94/20912

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 15/30	A1	(11) International Publication Number: WO 94/20912 (43) International Publication Date: 15 September 1994 (15.09.94)

(21) International Application Number: PCT/US94/02468

(22) International Filing Date: 4 March 1994 (04.03.94)

(30) Priority Data:
08/028,360 9 March 1993 (09.03.93) US

(60) Parent Application or Grant
(63) Related by Continuation
US 08/028,360 (CIP)
Filed on 9 March 1993 (09.03.93)

(71) Applicant *(for all designated States except US)*: C*ATS SOFTWARE INC. [US/US]; 1731 Embarcadero Road, Palo Alto, CA 94303 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: KLECKNER, James, E. [US/US]; 1855 Cowper Street, Palo Alto, CA 94301 (US). BECKSTROM, Rod, A. [US/US]; 1947 Emerson Avenue, Palo Alto, CA 94301 (US). GALVIN, Raymund, P. [US/US]; 755 Dora Avenue, Sunnyvale, CA 94087 (US). OGDEN, Sandy, L. [US/US]; 766 Palm Court, Sunnyvale, CA 94086 (US).

(74) Agents: McMAHON, Kevin et al.; Weil, Gotshal & Manges, Suite 280, 2882 Sand Hill Road, Menlo Park, CA 94025 (US).

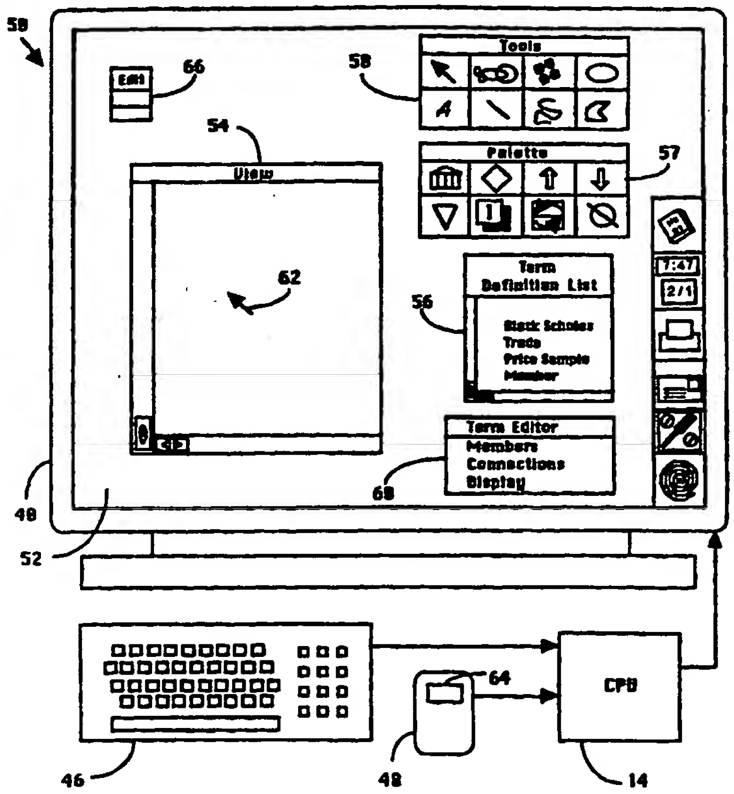
(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: AN OBJECT-ORIENTED SYSTEM FOR CREATING, STRUCTURING, MANIPULATING AND EVALUATING A FINANCIAL INSTRUMENT

(57) Abstract

A system for structuring and managing a financial instrument includes a financial framework comprised of a library of functions and data types supporting financial applications. The system enables a user to unambiguously define (60), price (712), and revalue (60) any arbitrarily complex instrument and manage the instrument together with other instruments in portfolios (56) to evaluate accounting profit and loss, capital requirements and costs, credit exposure, and to handle reporting requirements, manage risk, assist in meeting regulatory requirements, and produce all documentation relating to the instrument and transactions involving the instrument in a secure environment. Instruments, comprised of networks of interconnected terms, may form a part of other instruments.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic	SD	Sudan
CG	Congo		of Korea	SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

AN OBJECT-ORIENTED SYSTEM FOR CREATING, STRUCTURING, MANIPULATING AND EVALUATING A FINANCIAL INSTRUMENT

Background

The present invention relates to the field of finance, and within this field, more specifically to computer-implemented financial instrument and transaction structuring and management systems.

A "financial instrument" (or instrument) as used herein shall refer to the underlying structure of any financial transaction, such as a sale of goods or services, an investment device such as a bond, equity, foreign exchange contract or commodity, a currency, an index, derivatives of other instruments, such as swaps and options, etc. For a detailed discussion of derivative instruments, see Marki, Susan Ross, DERIVATIVE FINANCIAL PRODUCTS (HarperBusiness 1991), which is incorporated by reference herein.

Particular instruments, such as derivative instruments, are increasing in use at a rapid rate. For example, whereas, in the 1980s little, if any, of a bank's capital was tied to derivative instruments, today as much as one-half or more of a bank's capital may be tied to such instruments in complex structured investments and other transactions.

In general, a financial instrument represents a structured transaction, which may involve foreign currency exchanges, rate indices, contingent events, etc., and which may depend on the performance of one or a portfolio of other instruments. One of the powerful features of a financial instrument is that it can be used to "manage" aspects of the financial transaction it represents. Examples of such management include pricing, tracking performance, hedging, and monitoring related obligations such as coupon payments, etc.

Management of these instruments is currently handled principally by cash flow or parameter techniques, describing instruments as a collection of cash flows or as a set of parameters. For example, spreadsheets are commonly used to analyze or track the performance of an individual instrument. To analyze instruments whose performance is based on the performance of other instruments, the underlying instruments are first analyzed by way of a spreadsheet, then the evaluations are consolidated as required for the derivative instrument. For example, in order to evaluate the performance of a portfolio of instruments, the individual spreadsheets for the constituent instruments are consolidated together.

The use of spreadsheets is practicable only for a reasonable number of underlying instruments. For example, some banking institutions have a rule that spreadsheets can only be used for up to five transactions simultaneously. For more than this number of transactions, a custom software system must be used. However, an instrument may have any number of underlying instruments, and further those underlying instruments may themselves be comprised of other instruments. Thus, daily processing of portfolios, which may have hundreds or even thousands of underlying spreadsheets that may need to be aggregated to manage the portfolios, is essentially precluded.

In addition to the general solution of spreadsheets, specialized products and systems have been developed for various instruments and aspects of their management. Examples of such specialized products include SwapWare™ (for structuring and modeling interest rate and currency swap transactions, borrowings, and foreign exchange contracts), CapWare™ (for pricing and modeling caps, floors, collars, and forward rate agreements), Equity Derivatives™ (for structuring and modeling equity swap transactions), Strike™ (for structuring and modeling swaptions and over the counter bond options), etc., each sold by C•ATS Software, Inc., of Palo Alto, California. Elements of a portfolio, such as futures contracts associated with a portfolio, may be managed by Futures™, and reevaluation, accruals, cash management, duration, convexity, gap analysis, hedging, sensitivity and other analysis may be performed across the various specific products by Risk Manager™, also from C•ATS Software, Inc.

While these specialized products and systems are capable of addressing the specifics of particular instruments, they are not designed to be, and generally cannot be extended to handle new and different instruments without significant reprogramming and recompiling. Rather, to structure a new instrument (*e.g.*, define and instantiate its terms, performance characteristics, etc.), a new specialized system must be developed -- an expensive and time-consuming process. The ability to extend a system to accommodate new and different instruments shall be referred to herein as extensibility.

Furthermore, certain existing systems provide for evaluation of selected instruments, other systems provide for tracking the performance of instruments, and still other systems provide for monitoring obligations related to those instruments. While certain existing systems provide a number of capabilities, no single system which is presently available provides each of these aspects in an integrated fashion, reducing the communication costs for both humans and computers. The ability to tie these various aspects together shall be referred to herein as tight integration.

Another limitation in the art is the relatively high level of expertise in software development required for creation of new systems or modification of existing systems to handle new and different instruments. This has meant that the specialized products and services are developed either by an outside party for an investment organization, or by an investment organization's own software development group. Furthermore, modification of existing systems to handle new types of instruments has been the domain of third parties or in-house programmers. Investors have, for the most part, been barred for lack of substantive training and experience in computer programming from creating new systems or modifying existing systems for the purpose of structuring and managing new financial instruments. Since the ability of an investor to structure and manage a new instrument may differentiate that investor from other investors in the marketplace, ease of use and speed of development are at a premium.

The net result is that those in the financial field have not been provided with optimal tools for structuring and managing financial instruments.

Summary of the Invention

The present invention is a system for structuring and managing financial instruments. The invention includes a financial framework comprised of a library of functions and data types to support financial applications. It provides components from which processing capabilities can be built. The system is extensible, in that it can accommodate new and arbitrarily complex types of instruments, and tightly integrated, in that it can perform a variety of tasks with regard to the instrument in an integrated fashion. The system enables a user to unambiguously define, price, and revalue any arbitrarily complex instrument and manage the instrument together with other instruments in portfolios to evaluate accounting profit or loss, capital requirements and costs, credit exposure, and to handle reporting requirements, manage risk, assist in meeting regulatory requirements, and produce all documentation relating to the instrument and transactions involving the instrument in a secure environment. The instrument structuring and management system is extensible from one consistent framework, so that new instruments may be defined with established procedures and new procedures can be integrated with existing instruments.

The financial framework insulates all software tools from the particular computing environment and globally gathers basic functionality used in all tools. This means that implementation of the present invention is computer platform independent.

Three important services provided by the financial framework are: database management, update management, and a term evaluation engine. The database management service includes a data model, an Application Programming Interface (API), and data persistence, commonly referred to as data save-and-restore. The data model is organized hierarchically into four levels: Libraries, Definitions, Views, and Objects. Through these levels, the data model uniquely identifies each object in the system, allowing objects to be reused as needed. The API provides programmatic interfaces corresponding to each object and its relationships to the other objects in the data model, and allows new terms to be dynamically added to the system at run time. The API further allows external applications to be integrated with the present invention's financial framework so that existing applications and systems can work in concert with the present invention. Finally, the API also provides the ability to query the definition or composition of any data type at run time. Data persistence provides the ability to store and retrieve data in a database independent manner.

The update management service provides "links" between terms, enabling all data in the system to be updated. An object whose value depends on the value of other objects may be linked to the other objects so that when one of the other objects is modified, the linked object will be notified of the change so that its value may be appropriately modified (e.g., either dynamically or upon user request).

The term evaluation engine is a mechanism for evaluating a network of interconnected terms. All required term inputs must be evaluated before a term's outputs became valid (i.e., before its outputs can be computed). The evaluation is order independent for inputs, which means that evaluation may be apportioned. Each portion of the evaluation may be performed independently, for example using multiple processors (multiple processes), enabling significantly larger amounts of data and analysis to be handled than heretofore possible for financial instruments.

According to the present invention, an instrument is comprised of a plurality of interconnected terms. Terms are discrete objects which include at least a storage location, means for putting a value into the storage location, and an output function, and may further include other functions, one or more inputs, private data, etc. In certain embodiments, terms are encapsulated methods (*i.e.*, in an embodiment realized in an object oriented programming language environment). Terms may themselves be made up of other terms. The terms are connected together in a network such that appropriate output values from one term may be passed to another term as input values.

The terms which comprise an instrument are able to provide, together, sufficient data to allow the system to determine at least the Net Present Value (NPV) of the instrument with regard to a single currency, and may further provide data allowing the system to produce: NPV for any defined currency; partial derivatives such as delta (the first derivative with respect to price of a specified significant variable), gamma (the second derivative with respect to price of a specified significant variable), theta (the first derivative of price with respect to time), and vega (the first derivative with respect to volatility); a listing of payments; profit and loss information; mark-to-market accounting value; etc.

For each term which receives an input value from a term connected to it, the system automatically checks the type of input value to determine the permissibility of such a connection. The system alerts the user if an attempt is made to connect two terms where the required input value type is incompatible with the supplied output value type. For example, if a term which provides a date data-type output is connected to a mathematical term, such as a square root term that expects as an input a floating point number data-type, the system will alert the user that such a connection is impermissible, since the types are incompatible. Type checking of this sort is done only when a connection is established. Type checking of this sort is referred to herein as strong type checking.

Furthermore, certain terms are capable of various types of operations, and thus are capable of accepting various types of input values. For example, an "If-Then-Else" term may test the "if" condition to provide any data type as an output. With regard to these terms, the "if" condition input must evaluate to a Boolean, i.e., "true" or "false".

However, the output of the If-Then-Else term may be of any type specified by the "then" and "else" inputs, both of which must be of the same or compatible type. The system will alert the user if an attempt is made to connect an input value to the term which is incompatible with the required value type. Establishing the type of input and/or output values from a term as described above is referred to herein as inferred type checking.

A variety of methods may be employed to structure an instrument according to the present invention. For example, an instrument may be structured using graphical entry tools provided by a graphical user interface system, using a special Instrument Definition Language, using a spreadsheet system, using a parameter form entry system, etc.

In one embodiment employing a graphical user interface environment, a display window is provided, called a view window, in which a graphical representation of an instrument may be constructed. Another window, which contains a list of term definitions (referred to herein as a "term list"), provides a number of such term definitions from which users select. Associated with each term definition is a term graphic which is composed of an icon and a graphic display objects, such as a text field, etc. A term editor, which may be composed of a member editor, connection editor, display editor, etc., is used to configure the appearance of a term, to make connections between terms, and to edit values of the terms. In so doing, the software creates actual connections between terms, such as for passing values between terms. Thus, an instrument may be represented by a network of connected terms. Instruments constructed with the system may themselves by used as terms in the construction of other instruments.

Brief Description of the Figures

The present invention will be described in greater detail below, including specific examples thereof, with regard to the figures, in which:

Fig. 1 is an illustration of the general computer architecture of a system within which the present invention may operate;

Fig. 2 is an entity relationship diagram showing certain aspects of the architecture of a system according to an exemplary embodiment of the present invention;

Fig. 3 is an illustration of a number of user interface applications which may be used to compose definitions, and their relationship to the financial framework, according to one embodiment of the present invention;

Fig. 4 is an illustration of a term, member, and data structures for input, output, and data storage according to one embodiment of the present invention;

Fig. 5 is an illustration of the visual features and layout of a graphical user interface of an exemplary embodiment of the present invention;

Fig. 6 is an illustration of the relationship between the various instruments comprising one example of an embodiment of the present invention;

Fig. 7 is a table of price samples for past, present, and future performance of an equity forming a part of an example of an embodiment of the present invention;

Fig. 8 is an illustration of the state of a portion of the graphical user interface after certain steps are performed to assemble an equity instrument in an example of an embodiment of the present invention;

Fig. 9 is an illustration of a term graphic representing the equity instrument shown in Fig. 8;

Fig. 10 is an illustration of the state of a portion of the graphical user interface after certain steps are performed in the assembly of an equity spread instrument in an example of an embodiment of the present invention;

Fig. 11 is an illustration of the state of a portion of the graphical user interface after certain steps are performed in the assembly of an equity spread NPV term definition in an example of an embodiment of the present invention;

Fig. 12 is an illustration of a term graphic representing the equity spread NPV term definition shown in Fig. 11;

Fig. 13 is an illustration of the state of a portion of the graphical user interface after certain steps are performed in the assembly of an equity spread volatility term definition in an example of an embodiment of the present invention;

Fig. 14 is an illustration of a term graphic representing the equity spread volatility term definition shown in Fig. 13;

Fig. 15 is an illustration of a term graphic representing the equity spread instrument shown in Fig. 10;

Fig. 16 is an illustration of the state of a portion of the graphical user interface after certain steps are performed in the assembly of an equity spread call option instrument in an example of an embodiment of the present invention;

Fig. 17 is an illustration of a term graphic representing the equity spread call option instrument shown in Fig. 16;

Fig. 18 is an illustration of the state of a portion of the graphical user interface after certain steps are performed in the assembly of an Ajax-Monolithic equity spread call option instrument in an example of an embodiment of the present invention; and

Fig. 19 is an illustration of the method for accessing external terms according to one embodiment of the present invention.

As between each of these figures, like reference numerals shall denote like elements.

Detailed Description

For purposes of illustration, the present invention will be described by way of selected examples (or applications), with reference to a number of embodiments of the present invention where appropriate. It will be apparent to one skilled in the art that the examples, and the environment in which they are developed, demonstrate the functionality and features of the present invention. It will also be apparent to one skilled in the art that certain variations, additions, deletions, modifications, etc., to the examples are contemplated, and that recitation of these examples does not limit the scope of the invention.

Fig. 1 illustrates the general architecture 10 of a system of the type within which the present invention operates. Architecture 10 comprises a main bus 12, which serves to interconnect various components, including at least some of the following: Central Processing Unit (CPU) 14, Floating Point Unit (FPU) 16, Integrated Circuit Processor (ICP) 18, Bus Controller 20, Video RAM 22, Dynamic RAM (DRAM) 24, Static RAM (SRAM) 26, Digital Signal Processor (DSP) 28, Internal Hard Disk 30, External Memory Device 32 (connected for example via a SCSI port 34), External Network Devices 36 (communicating for example over an Ethernet Network 38, and connected via SCSI port 34), Display Device 40 (such as a CRT), Printer 42 (such as a PostScript device connected for example via a serial port 44), Keyboard 46, and Pointing Device 48 (such as a mouse, trackball, etc.) Although each of the above elements are well known to one skilled in the art, the function and significance of particular components will be discussed in further detail where appropriate.

Many modern computer systems employ all-points-addressable (APA) or bit-mapped display screens which support a graphical user interface (GUI). Popular GUI platforms include the Apple Macintosh series of computers, personal computers running the Microsoft Windows software package, and the NeXT system, manufactured by NeXT, Inc. of Redwood City, California. In these systems, a user interacts with the computer using a keyboard and pointer control mechanism such as a mouse. Generally, the result of a user's interaction with a software application is displayed in a region of the screen referred to as a window. Software application packages, documents, files and the like may be represented as icons, and launched or opened by aligning the pointer over the icon and double clicking a button on the mouse.

Focusing on the user interface for the NeXT system, it consists of two elements, NeXTStep and Display PostScript. The NeXTStep element itself consists of four components: the Workspace Manager, the Interface Builder, the Applications Kit, and the Window Server. The Workspace manager is the user interface component which displays files and directories, manages the icon dock (a region of the display screen in which icons representing application software are located), and launches applications. The Interface Builder is the user interface component which allows an application designer to lay out the application's user interface. The Application Kit is a software library of graphical objects such as windows and buttons. The Window Server is the user interface component which manages the display screen, handling drawing requests for the various running programs, and determining which programs will be notified about events such as mouse clicks and key presses. The Window Server in turn calls Display PostScript to perform any actual drawing required, either for the display or for printing. (For a more detailed description of NeXT Step and its related elements, see Webster, Bruce F., THE NeXT BOOK (Addison-Wesley 1989) which is incorporated by reference herein.)

With reference now to Fig. 2, there is illustrated therein an entity relationship diagram 500 showing a high-level view of the organization of data within one embodiment of a system according to the present invention. Entity relationship diagrams are commonly associated with object oriented software systems, although the architecture represented by such a diagram may be applied to other systems. The boxes shown in Fig. 2 correspond to entities.

An entity relationship diagram provides details about the high level data organization of the software system. One implementation method is the use of classes of the C++ programming language to represent entities and pointers or pointer-like data types to represent relationships. As well understood by one skilled in the art, programmer-defined classes provide data abstraction by allowing the representation details of objects to be hidden and accessed exclusively through a set of operations defined by the class. (See, e.g., Ellis, Margaret and Bjarne Stroustrup, THE ANNOTATED C++ REFERENCE MANUAL (Addison-Wesley 1990) and Coplien, James O., ADVANCED C++ PROGRAMMING STYLES AND IDIOMS (Addison-Wesley 1992), each of which being incorporated herein by reference). An example of the entity structure shown in Fig. 2 is that all entities are derived from and inherit the properties of Entity entity 502. Entity entity 502 may provide, for example, the methods for objects to be created, the strategy for dynamic memory allocation and deallocation, the storage and retrieval methods, etc.

It will be appreciated that the user interface according to the present embodiment may be event driven. An event is a significant, asynchronous change in the system that requires processing. An example of an event is the click of a button on a pointing device such as mouse (not shown). The user interface is "driven" by events since user and the system interface via events. The evaluation engine described below is, however, demand driven. That is, an event initiates a system action, and there are no further events until the system returns a value, which may be after significant processing activity by the system. During evaluation, terms of the system "demand" a value from other terms for their own processing when needed. Update notification is event driven, and notice of updates to values is delivered to each term which employs that value. It will be further appreciated that in appropriate contexts, the evaluation itself could be event driven, in which case Entity entity 502 may provide, for example, the methods for event handling, and further that the interface need not be event driven.

In Fig. 2, the "=" sign denotes an "is a" relationship. An "is a" relationship means that the definition of an object of one class *is a* variation of the definition of objects of another class. (The "is a" relationship is similar to the concept of inheritance in languages such as C++, but implies less detail about the nature of the objects and relationships between classes.) The entity closest to the "=" corresponds to the base entity from which the other entities inherit information, the entity (or entities) farthest from the "=" is (are) the derived entity (entities). For example, the Library entity 504, is an Entity entity 502. That is, all members of the Library class are also members of the Entity class. The same is true for the Definition entity 506, View entity 508, and Object entity 510. Each member of these classes is also a member of the Entity class.

An arrow with a single arrowhead shown in Fig. 2 denotes a singular relationship. For example, the single arrow pointing from the Entity entity 502 to the Definition entity 506 means that there is only one definition of the Entity entity 502 (*i.e.*, each entity is uniquely defined). An arrow with a double arrowhead denotes a plural relationship. For example, the double arrowhead side of the arrow pointing between the Library entity 504 and the Definition entity 506 means that the Library entity may have multiple definitions.

A single or double arrow may have an optional "O" or "|" associated with it. The "O" (for optional) means that the relation may have zero entities and be empty. The "|" implies that the relationship must have exactly one entity in the relationship. For example, the single arrowhead side of the arrow pointing between the Library entity 504 and the Definition entity 506 has a "|" associated with it, meaning that there is always at least one Library entity 504 for a Definition entity 506 (*i.e.*, a definition always resides in exactly one library).

The system according to the present invention implements a run-time type system similar to that of Smalltalk (see, e.g., Goldberg, Adele SMALLTALK-80: THE INTERACTIVE PROGRAMMING ENVIRONMENT (Addison-Wesley 1985)), or Objective C (see, e.g., Pinson, Lewis J. and Richard S. Wiener, OBJECTIVE-C OBJECT-ORIENTED PROGRAMMING TECHNIQUES (Addison-Wesley 1991)). A run-time type system uses a special object to describe the types of data in the system and permits some common operations on the data such as create, delete, copy, etc. This special object is sometimes called a "metaclass" object. In this system it is called a "Definition" object. The definition is also used to access type-specific information such as sub-types contained in the type, functions (or methods) that operate on the type, and input data needed by the functions.

A number of interface applications may be used to compose definitions. Fig. 3 illustrates several such applications, including graphical editor 602, spreadsheet application 604, Instrument Definition Language (IDL) interface 606, and form entry interface 608. Interface with a Framework Programming Interface (FPI) 610 of the financial framework is accomplished via an appropriate compiler 612, 614, 616, 618. The financial framework interfaces to the database 620.

The Application Programming Interface (API) specification for C++ classes that implement the data model of Fig. 2 is detailed in Appendix A attached hereto. Also, attached hereto as Appendix B is a description of data structures suitable for implementing the mechanism according to one embodiment of the present invention. In each of these appendices, it will be appreciated that the prefix "Cats" specifies the software package, and forms no other part of the code, instructions or data listed therein. Thus, the prefix does not form a limitation of the API and structures.

Each definition contains a collection of "named" View entities (views) 508. Names of objects are similar to path names, and follow the data organization model discussed above (Fig. 2). For example, the name Library_A:Definition First:View_Interface:Object_Payment_List refers to the object named "Object_Payment_List" in the view named "View_Interface" in the definition named "Definition_First" in the library named "Library_A". The name of an entity uniquely defines it within its scope. Thus, two libraries may not have the same name, two definitions within the same library may not have the same name, two views within the same definition may not have the same name, and two objects within the same view may not have the same name. Each view has a type, for example interface, model, icon or editor. An interface view of a definition declares the definition's Application Programming Interface (API). Therefore, in this embodiment of the present invention, each definition is required to have an interface view, and there is permitted only one interface view for a definition. It is through the interface view that the input and output members of a definition can be accessed.

Each view consists of a collection of named Object entities (objects) 510. Object names are unique within the scope of their containing view. There are three major subclasses of objects within a view: Term entities (terms) 512, I/O entities (I/O) 514, and Graphic entities (graphics) 516. The primary building blocks for structuring an instrument are terms. Terms may include one or more data buffers that are declared by Member entities (members) 522 in the definition of the term (as found by following the "is a" relation on the object base class).

As used herein, an input or output is the combination of a term and a member, marked as input or output in the definition of the term. Thus, a member may be used as an input, as an output, or as both input and output. In addition, a member may be used as a storage location (or assigned a value). Input and output members declare the public interface of a definition. When a definition is instantiated as a term, the public interface of that definition becomes the inputs and outputs of the term. All members marked as input and/or output of a single definition are contained in the interface view for that definition for convenience of access in the current embodiment.

With regard to Fig. 4, an example of connecting the input of one term to the output of another term is illustrated. Both terms reside in the view Library: Example:Interface 702. The term named "ToTerm" 706 has one input 710 that is connected to the term named "FromTerm" 704. FromTerm 704 is an instance of the definition Library:FromTerm (not shown) that is created by the view Library:FromTerm:Interface 720. Library:FromTerm:Interface 720 contains exactly one member 722 named "OutMember" that is marked as an output member by setting an IsOutput value for it to True.

Similarly, ToTerm 706 is an instance of the definition Library:ToTerm (not shown) that is created by the view Library:ToTerm:Interface 730. Library: ToTerm:Interface 730 contains exactly one member 732 named "InMember" marked as an input by setting an IsInput value for it to True.

A connection is made between FromTerm 704 and ToTerm 706 using the input data structure 710. Input data structure 710 is accessed from ToTerm 706 and contains pointers to FromTerm 704, OutMember 722, InMember 732, and a value buffer 712 for holding data of the type of the connection.

As stated above, the evaluation of the present embodiment is demand driven, meaning that a term will only receive an input if and when it "demands" it from another term. For the purpose of explanation, assume first that Term 706 demands a value from a term 704. Appropriate methods for this demand will be part of the Definition of view 720 from which Term 706 is instantiated. Input data structure 710 contains the address of the "From" term 704. This is used by Term 706 to identify the term providing the demanded value. Input data structure 710 contains a pointer to OutMember 722 of the From term that returns the demanded value. (Together, term 704 and Member 722 form an "input object" 710 containing all information needed to obtain data from another term.) Finally, the manner in which the From term 704 provides the value to Term 706 is that as part of the demand for a value, Term 706 provides its own memory location into which the From term 704 will write the value. This memory location is identified by Value Buffer 712. The value of the FromTerm 704 is thus made available to ToTerm 706 by looking up the GetFunc 742 function and calling it with the arguments of FromTerm 704, OutMember 722, and value buffer 712. The GetFunc 742 function proceeds to evaluate any inputs in a similar fashion (none in this example) and then stores the proper value into value buffer 712 and returns.

The "get function" stored in GetFunc 742 for a compiled term can be implemented as a member function of its associated class. As a result, the OutMember 722 argument need not be consulted since the proper offset into the term's data structure is provided by the C++ compiler. For structurally composed terms, the OutMember 722 causes the FromTerm 704 value to be stored on the definition of FromTerm (not shown). If there are input members (not shown) to the structurally composed term, they consult the pointer stored on the definition to find and access the particular input data for that term (in this example the term would be FromTerm 704). The found input data structure is then in turn evaluated, if necessary, and the data is returned by the input member (or error if not available).

Furthermore, a term will keep a list, as FromTerm.Outputs 708, of all terms which could demand its value. That is, each time a definition is instantiated as a new term which could demand a value from a supplying term, the new term's "interest" in the value of the supplying term is registered by adding an identification of the new term to the supplying term's Out_list. It will be appreciated that in other embodiments it may desirable to automatically write a value into all interested terms, which is facilitated by a term's Out_list. For example, when a value of a term is updated, the updated value may propagate through the system. Alternatively, an update to a value may result in a notice of the update being provided to interested terms, which may then request the updated value if and when appropriate. Thus, the Output data structure 708 may perform a substantial role in the update function.

Input data structure 710 may perform one other function, which is to serve simply as a data storage location for Term 706. In this case, the pointers other than the value buffer 712 are not used. In all three cases (input, output, and storage), the memory space of the unused components may be captured for other purposes, thereby reducing required memory space.

A network of interconnected terms forms an evaluation network. An input of a term may be connected (via a "Connect()" function) to the output of any term contained within any of the views of the same definition, provided that the "signature" or type of the input and output are "compatible." The signatures are compatible if the type of the data on the input is equal to or is a subclass of the type of the data on the output. It is important to recognize that connections are not made between definitions. Rather, each definition can be instantiated (as a term) within a view (or views) of a new definition, and the connection is made between terms within the scope of the new definition.

One additional aspect of the system illustrated in Fig. 2 merits mention. The dashed box 520 labeled "Connection" represents a class that is used to simplify the specification of several complex relationships among members and terms. The connection class is not an entity, and is not itself stored in the system's "persistent store." It is typically allocated as a temporary variable in a subroutine and used either to query about the existence of a connection between terms or to iterate over those connections.

Other details of the system architecture will be evident to one skilled in the art given the definitions and architecture of Fig. 2. Furthermore, it will be evident to one skilled in the art that the above description is only one embodiment for implementing the architecture described in Fig. 2, which itself is only one embodiment of the present invention. For purposes of illustration, however, the implementation of the entity relationship diagram 500 of Fig. 2 will be described in detail where appropriate in the following example of the structuring and management of a financial instrument according to the present invention.

With reference now to Fig. 5, there is illustrated therein the visual features and layout of a graphical user interface 50 of an exemplary embodiment of the present invention. The graphical user interface 50 of this embodiment comprises a number of windows displayed in a workspace 52 on a display device such as the display device 40 discussed above.

The first window displayed in workspace 52 shown in Fig. 5 is the view window 54 in which term definitions and instruments are structured and displayed. As stated above, instruments are comprised of terms, and a term definition list containing a number of definitions of terms from which a user may select to build an instrument are displayed in the Term Definition List window 56. A number of icons representing terms or instruments, either precompiled or created by the user, may also be provided to simplify the instrument structuring process. These icons are presented in a palette window 57. A selection of tools are also provided for the construction, as well as the display of the visual aspects of an instrument, related text, and graphical objects, similar to well-known computer drawing programs. These tools are displayed in the tool window 58. Finally, a user may view and edit a term during or after construction using various editors, such as a member editor, connection editor, and display editor. Selection of the editor is made in the Term Editor window 60.

Typical for graphical user interface systems, "selection" of objects such as terms, tools, editor, etc., is accomplished by positioning the pointer 62, using a pointer device 48 (such as a mouse) and actuating (clicking) a button 64 located on the pointer device 48. In addition, one or more pull down menus, such as menu 66, are provided in appropriate locations on the display. Alternatively, keyboard "shortcuts" for certain selections, such as "CMD S" (actuation of the "command" and "S" keys simultaneously) which performs the save command, are provided.

An example of instrument creation will now be given, together with pertinent system details. The instrument of this first example will be a particular type of option known as a spread option. The instrument underlying this type of option is not a specific, single security, but rather the value of a relationship, for example between two different securities. Spread options are often used for hedging; that is, for securing a backup position in the event that one investment fails to be profitable. For a more detailed discussion of spread options, see Marki, Susan Ross, DERIVATIVE FINANCIAL PRODUCTS, pp. 80-81 (HarperBusiness, 1991).

The particular option that shall be structured in this example is an equity spread option between the price of Ajax Corporation shares and Monolithic Corporation shares. For the purposes of explanation, we will assume that Ajax Corporation is primarily in the hardware business, and Monolithic Corporation is primarily in the software business. This spread option might then represent an investment based on the belief that hardware will be more profitable than software (or vice versa). The option is a call option, meaning that it is an option to buy shares. For convenience, the option of this example shall be referred to as the "AMESCO" option, for Ajax-Monolithic Equity-Spread-Call-Option. Fig. 6 illustrates the AMESCO option, with reference to other figures where appropriate. The structure of instrument 550 of this example is an equity-spread-call-option trade definition 240, which contains instances Ajax 130 and Monolithic 131 of Equity Instrument definition 556, an instance Equity_Spread 190 of Equity Spread Instrument definition 150, and an instance Equity_Spread_Call_Option 230 of Equity Spread Call Option Instrument definition 200. The Equity Spread Instrument Definition 150 contains an instance Equity_Spread_Volatility 155 of the definition Equity Spread Volatility 154 and an instance Equity_Spread_NPV 180 of the definition Equity Spread NPV 152. The Equity Spread Call Option Instrument definition 200 contains an instance 224 of the definition Black-Scholes 551.

The Ajax_equity_instrument term 130 and the Monolithic_equity_instrument term 131 will each provide: (1) their Net Present Value (NPV); (2) volatility; and (3) instrument name. Volatility is the measure of the expected standard deviation of a price of an instrument in one year with respect to a significant variable, for example, the expected standard deviation of the price of an Ajax equity share one year from the present date, in U.S. dollars.

In order to determine the NPV, the equity instrument must obtain either a "fixed" price for a particular instrument on a particular date, or a calculated (or forecast) price of the instrument on a particular date. "Fixed" prices are provided if the date for which the NPV is to be obtained is a date for which the system has access to an actual closing price for the equity. Otherwise, calculated prices are provided. This may be visualized with regard to Fig. 7, which shows a table 700 of Ajax closing prices. The price of an Ajax share is fixed at the close of business each day. Prices in the table for past share closing-prices, or actual past performance, of Ajax shares are fixed prices shown in region 702 of table 700. Therefore, price samples in the table for dates prior to the present date generally return a fixed value as indicated by a "Y" (for yes) in the Fixed? column. These fixed values may be entered manually, or may be acquired from a real time data feed service 704. It is also possible to associate an identification of the person fixing a price with each fixed value, such as the data of the Fixer column. Alternatively, a user may be provided with the ability to manually enter a price for the equity in order to simulate an event or model an instrument's performance for evaluation, etc.

For the present date 706, and before the close of business for that date, the value of the equity is not yet fixed. Thus, for the present date, the value may be the actual present value, since the data may be returned from a real time data feed service or otherwise entered once acquired, or may be calculated if there is insufficient access to the actual value.

Price samples for dates other than those which the system has actual historical values for will return a calculated price. For example, future performance will be based on forecast prices as shown in region 708, and calculated past performance is shown in region 710. The actual manner in which the price sample is determined is beyond the scope of this disclosure. However, it is sufficient to note that the algorithm or other methodology for calculating the value may form a portion of the definition of the price sample term, so that the value may be calculated based on inputs from other terms, or the value may be predetermined and manually entered by the user either into a table such as that shown in Fig. 7 or entered directly as an input into the equity instrument 556 (Fig. 6). In either case, entry of the appropriate value into the price sample term may be handled by way of an interactive editor portion of the system.

Returning briefly to Fig. 5, there is illustrated therein the initial state of the system for the structuring of the AMESCO option. The view window 54 is initially blank, and the term list, palette, tools, and editor windows 56, 57, 58, and 60 are displayed.

With reference now to Fig. 8, there is illustrated therein the state of the system after several steps in the process of structuring the AMESCO option have been performed. The first step in structuring the option is to construct the two underlying equity instruments 556 and 558. A view window 100 is created for the Ajax equity instrument. It is assigned the name Demo:Equity_Instrument:Interface according to the syntax discussed above. A number of terms are then selected from either the term list window (not shown) or the palette (also not shown), and graphic display objects or icons representing those terms are located in the view window 100.

Fig. 8 shows a number of term graphics representing terms placed in view 100. These term graphics represent a Curve_import term 102, a Volatility term 104, and a Curve_rate term 106. Also in view 100 are icons for input members Curve_name 108, and Date 110, and output members Instrument_name 112, Volatility 114, and Equity1_NPV 116.

One form of a term graphic is a rectangular region containing a text portion for displaying a term name, a display portion for displaying one or more of the term's values, and an interactive portion, such as a slider control, buttons or the like for modifying the term's values. Aspects of the creation of the interface of the system, such as creation of these term graphics, may be handled by an interactive display editor portion of the system, while the user's interaction with that interface is handled by the system itself (with certain exceptions such as text editing, which may be passed to a text editor). In one embodiment of the present invention, the user is provided with the ability to control which values are displayed by a term graphic. For example, a "switch" may be provided in the members portion of the Term Editor window 60 (shown in Fig. 5) for selecting the individual values of a term, its inputs and outputs, etc., for display.

Once the members and terms are instantiated in view 100, they may be interconnected to form the equity instrument, instances of which are Ajax equity instrument 130 and Monolithic equity instrument 131. According to one embodiment, the user initially selects the connection view option from the Term Editor window 60 (shown in Fig. 5), for example by way of a pull down menu. The user next selects a term or member which is to receive an input, for example by positioning the pointer over the term and clicking the mouse button. The connections view in the Term Editor window 60 will then provide a list of the possible inputs to that term or member from which the user may choose. Once the input is selected, the user selects the term or member for providing the output. While holding down an appropriate key or the mouse button, the user moves the mouse to position the pointer over the term or member providing the output to be connected to the selected input, then releases the button or key. This causes a graphical connection 118-128 to be established and displayed between the terms and/or members.

Upon establishing the graphical connection, an underlying connection is established between the two terms in software. This underlying connection can serve a number of purposes, but in its most basic form it is a call by one term to another term, for example to pass a value between the terms. Thus, not only is a visual connection established between the terms, data is caused to be written to support an underlying physical interconnection (via software) between the terms. Terms may also be connected by referencing the name of the term, for example by using the term's name in an equation. This type of connection is referred to as a connection by name, an example of which is provided below. Once the connection is established, the user may easily change the terms or members to the connection by simply repeating the above process.

Once the interconnections are established between the input members and the proper terms, the terms may use input data in performance of their functions. For example, Curve_import term 102 inputs a curve name and, based on the curve name, reads a file representing a curve. The file may be the file illustrated in Fig. 7. The data read into memory by the Curve_import term 102 is then made available to the Curve_Rate term 106. Although the file contains data representing actual closing prices of the equity, that data may be used by the Curve_rate term 106 to calculate the values of the prices for dates other than those stored in the file. For example, interpolative algorithms (or other techniques) may be used to determine the closing price of the Ajax equity for dates earlier than the earliest available actual closing price, or to predict the closing price at some date in the future. Thus, the NPV may be determined by the appropriate method by the Curve_rate term 106 by utilizing the data provided by the Curve_import term 102 and the date provided by the Date member 110.

Likewise, the volatility may be calculated by the Volatility term based on the data provided by the Curve_import term 102. Formula for calculating the volatility are well known in the art. See, for example, Figlewski, et al., FINANCIAL OPTIONS: FROM THEORY TO PRACTICE (Business One Irwin 1990), which is incorporated herein by reference. As an alternative to calculating a term's value, for example volatility in the case of the Volatility term 104, that value may be entered manually, or imported from another system or routine. In the case where the value is entered manually, that entry may be by way of an input member, as previously discussed, or may be by way of private data as discussed in further detail below.

With regard to the case of importing the value, it will be appreciated that means for conversion of data from one format to another are well known, and that importing data is, in significant part, converting data from one format which is not usable to another format which is usable. Furthermore, it is possible according to the present invention to call and run an external routine (or term) and import the data returned by that external routine (or term).

For simplicity in later use of the equity instrument 556, it may be instantiated as a term using a term graphic composed of an icon of the user's choosing, for example as stored in a TIFF file or the like, together with a graphic display object (such as a text field, slider control, etc.) (This graphic display object is one type of graphic entity 516 shown in Fig. 2.) Again, using the Term Editor window 60, this time in the display editor for an instance (term) 130 of the equity instrument the desired term graphic representation of the instrument is selected. This may be displayed as term graphic 130 shown in Fig. 9. Term graphic 130 may be given a unique name 113, such as "Ajax", and certain of its values displayed as appropriate, for example the outputs volatility 114, NPV 116, Instrument_name (not shown), and "isa" 112 (which is the type of every entity and thus term as well).

In setting up the AMESCO example, there are two underlying equity instruments. The first is the Ajax equity instrument 130 described above, and the second is the Monolithic equity instrument 131. It will be appreciated that the Monolithic equity instrument term will be created in the same fashion as the Ajax equity instrument term 130, and represented by a term graphic 131 (shown in Fig. 18), and therefore the details of creating that term are not discussed further herein.

The next instrument that is structured according to the present example is an Equity_spread instrument 150 shown in Fig. 10. This instrument is itself composed of two terms, an Equity_spread_NPV 180, which is an instance of the Equity_spread_NPV definition 152 and Equity_spread_volatility 155, which is an instance of the Equity_spread_volatility definition 154. Equity_spread term 150 also includes input members Equity1_NPV 156, Equity2_NPV 158, Equity1_ volatility 160, Equity2_volatility 162, and Covariance 164. Finally, Equity_spread_instrument 150 includes output members Instrument_name 166, NPV 168, and Volatility 170.

With regard to Fig. 11, there is shown in detail therein Equity_spread_NPV term definition 152. Equity_spread_NPV term definition 152 consists of a DoubleMath2In term 172, input members Equity1_NPV_in 157 and Equity2_NPV_in 159, and output member NPV_out 174. Input member Equity1_NPV_in 157 is connected to the Equity1 _NPV input term 156, shown in Fig. 10, which in turn is connected to NPV output term 116 of Equity_Instrument 556, shown in Fig. 8 for the term Ajax 130 in AMESCO 240.

Input member Equity2_NPV_in 159 is connected to Equity2_NPV term 158, shown in Fig. 10, which itself is connected to a corresponding NPV output member 166 in the Equity_Instrument 556 for the Monolithic term 131. DoubleMath2In term 172 is an instantiation of a generic math term and the "subtract" output is connected to the NPV_out output member 174. As its name implies, DoubleMath2In term performs its subtract function on 2 values, A and B, each value being a double precision floating point number. It yields the value of output member NPV_out 174, which is also a double precision floating point number, and which is connected to NPV term 168 shown in Fig. 10. As shown in Fig. 11, DoubleMath2In term 172 displays the values of A, B and the result of the subtraction. The Equity_spread_NPV term 152 is represented by a term graphic 180, displaying the instrument's name and the value of the NPV output member 174, as shown in Fig. 12.

The second term which comprises Equity_Spread_Instrument 150 is Equity_Spread_Volatility 154. It is created in much the same fashion as Equity_spread_NPV term 152, but with one distinction worth mentioning. As shown in Fig. 13, the Equity_Spread_Volatility instrument 154 consists of Formula term 182, input members Equity1_volatility_in 161, Equity2_volatility_in 163, and Covariance_in 166, and output member Volatility_out 169. Equity1_volatility_in term 161 is connected to Equity1_volatility term 160, shown in Fig. 10, Equity2_volatility in term 163 is connected to Equity2_volatility term 162, shown in Fig. 10, and Covariance_in term 166 is connected to Covariance term 164, also shown in Fig. 10. Likewise, Volatility_out term 169 is connected to Volatility term 170. As shown, Formula term 182 is defined with reference to variable names, such as Equity1_volatility_in, Equity2_volatility_in, and Covariance_in. By defining a term in this manner, connections are made between the input members and the terms by using text names. Graphical connections described above need not be made. For comparison, a graphical connection 184 is shown in Fig. 13 between formula term 182 and output member Volatility_out 169. Once again, this instrument may be represented by a icon 155, shown in Fig. 14. (It should be noted that member names alone do not imply a connection. For example, members with the same name, such as NPV output term 168 of Fig. 10 and NPV input term 202 of Fig. 16 are not necessarily connected, and in fact are only connected when they are connected as described above.)

Returning to Fig. 10, the two terms Equity_spread_NPV 152 (represented by term graphic 180) and Equity_spread_volatility 154 (represented by term graphic 155) are shown connected to the input and output members. Equity_spread instrument 150 may, itself, be represented as a term graphic 190 together with a display of desired values such as NPV 168 and Volatility 170, as shown in Fig. 15.

The final instrument which needs to be structured for the AMESCO example is the Equity_spread_call_option instrument 200, shown in Fig. 16. The Equity-spread_call_option instrument 200 consists of input members Underlying_price 202, Strike_price 204, Volatility 206, Risk_free_rate 208, Value_date 210, and Expiration_date 212, and output members NPV 214, Delta 216, Gamma 218, Theta 220, and Vega 222. These output members correspond to the outputs of a term which implements the Black-Scholes option pricing model 551, the Black_Scholes term 224.

As a slight variation on the methodologies presented above, selected values used by the Black_Scholes term 224 may be part of the definition of the term itself, as opposed to an input. For example, in the process of instantiating the definition as a term, the user is provided with the opportunity to enter values into the term via the members editor for the term. Data entered in this fashion is referred to as private data, since it is not readily apparent to the user. Private data may be "locked" into a term. Those with proper authority may be permitted to access and modify the data, while those without the proper authority may only view the term. This provides both a security function and data integrity.

In a manner similar to that described above, the Equity_spread_call_option instrument 200 may be instantiated as a term and represented by a term graphic 230 together with selected values such as those of the output members, as shown in Fig. 17. The completed AMESCO instrument 240, comprised of the underlying terms and instruments represented by their term graphics 130, 131, 190, and 230, is shown in Fig. 18. It will be appreciated that the AMESCO instrument 240 may itself be represented by a single term graphic (together with appropriate values) and used in the structuring of another instrument, for example representing an option on this option or the like (not shown).

Instruments structured by the above process and using the system of the present invention may be managed in a multitude of ways. For example, given a large number of instruments such as a portfolio of AMESCO instruments structured per the above discussion, it may be desirable to produce a list of obligations, resets, payments, risk measures or other attributes of the instruments. This would be facilitated by the present invention by building into the definition of each instrument Make_list terms which produce lists of desired objects or attributes for that instrument. In turn, when defining the portfolio instrument, a user would employ a Make_list term to produce a "master" list of the desired objects or attributes from the lists produced by each instrument's Make_list. An additional term may then be used to "filter" or sort the master list to produce subsets of the attributes, for example all obligations due on or after a certain date, etc.

Another aspect of the present invention is the ability to export and import data from the system for structuring and management of a financial instrument (importing data has been addressed briefly above). The Integrated ASCII import/export mechanism allows data from the system's database to be imported and exported from/to an ASCII file. This allows the data to be employed in existing financial packages. For example, the profit or loss, capital requirements, credit exposure, etc. of individual instruments or portfolios of instruments may be managed by the aforementioned Risk Manager™ from C•ATS Software, Inc.

Specific to this mechanism is the format of the file. The data and the label identifying the value are placed on the same line (separated by a user specified delimiter) forming a record.

Within the file, the records are organized into logical groups. The data within one group forms a unit within the system's database (e.g., a cashflow, a trade). The groups are identified by a start and end label of the format BEGIN XXXX and END XXXX, where XXXX is the group identifier.

Examples of the system's databases which can be imported and exported in this format:

+ Transactions
+ Institutions
+ Banks
+ Quotes
+ Market data curves

Multiple items (e.g., transactions) can be placed in one file with the appropriate group identifiers being repeated for each transaction. Also within an item (e.g., a transaction) some records can be repeated (e.g., cashflows).

A list of records is printed in the ASCII IMPORT/EXPORT user manual for one implementation of the present invention offered by C•ATS software, and which is incorporated by reference herein. Within the groups there are certain key fields which, according to this embodiment, must always be present within the group.

The group identifiers (and keys) are:

Transactions:

BEGIN TRANBASE
END TRANBASE

Main part of the transaction
Keys: Code and Leg

BEGIN PL
END PL

P&L records (can be repeated)
Keys: Code, Leg and Date

BEGIN SIDEBASE
END SIDEBASE

Information about the sides (up to three depending on type of transaction)
Keys: Code, Leg and Side

BEGIN CASHFLOW
END CASHFLOW

Cashflow records (can be repeated)
Keys: Code, Leg, Side, Coupon Date, Period Date and Type

BEGIN RESETPARM
END RESETPARM

Reset parameters

Keys: Code, Leg, and Side

BEGIN RESETREC

END RESETREC

Reset records (can be repeated)

Keys: Code, Leg, Side, Reset Date, Effective Date and Period

<u>Date</u>

BEGIN TRADEBASE

END TRADEBASE

Trade parameters

Keys: Code, Leg, Definition Code and Definition Leg

BEGIN TRADEREC

END TRADEREC

Trade records (can be repeated)

Keys: Code, Leg and Trade Id

<u>Institutions</u>

BEGIN INSTITUTION

END INSTITUTION

All information about the institution

Key: Party

<u>Banks</u>

BEGIN BANKACCOUNT

END BANKACCOUNT

All information about the bank

Key: Bank Id

<u>Quotes</u>

BEGIN QUOTE

END QUOTE

All information about the quote

Keys: Date, Term, Currency, Option, Source and References

<u>Market Data Curve</u>

BEGIN DISCBASE

END DISCBASE

All information about the discount curve

Keys: Currency and Initial Date.

Another example is the SQL I/O, which allows a user to write data from the system to an SQL database. This is accomplished by writing an SQL statement from the system.

<u>Inputs</u>

128 inputs that can be optionally attached to outputs of terms or have string data entered.

<u>Outputs</u>

An SQL I/O is provided which evaluates and prints each input as an SQL string to a file which may be sent to a database, such as a Sybase database. A typical path name used by one embodiment of the present invention is:

```
ProgName|INSERT        <table name>
                           (<field1>,<field2>, . . .)
               VALUES      (value1, value2, . . .)
```

Figure 19 details the calling of external functions (e.g., main, executable) from an external source. For example, programs may be written in most modern programming languages such as C, C++, FORTRAN, as shell script programs, etc., which augment the capabilities of a system according to the present invention. The method for accessing these external functions is as follows. In Fig. 19, an instrument 300 has been partially created according to the above description. Instrument 300 includes an external function call term 302, which includes an output 304 and an input member 308. Term 302 further includes an identifier 312 of the external function to be called, such as the pathname of the routine. The values are passed to the external program 314 as parameters, and the results from the external program 314 are returned to the external function call term 302.

It will be appreciated that the present invention enables a user to unambiguously define, price, and revalue any arbitrarily complex instrument and manage the instrument together with other instruments in portfolios to evaluate accounting profit or loss, capital requirements and costs, and credit exposure, and to handle reporting requirements, manage risk, assist in meeting regulatory requirements, and produce all documentation relating to the instrument and transactions involving the instrument in a secure environment.

In general, to those skilled in the art to which this invention relates, many changes in construction and widely differing embodiments and applications of the present invention will suggest themselves without departing from its spirit and scope. For example, the above description has been in terms of structuring and managing a financial instrument. However, it will be appreciated that application of the present invention in not necessarily limited to financial instruments. Furthermore, applications other than structuring and management may be performed by the present invention, for example allowing a user to simulate the performance characteristics of an instrument without requiring a detailed knowledge of the components of the instrument (similar to the idea of "reverse engineering" of hardware or software). Thus, the disclosures and descriptions herein are illustrative, and are not intended to be in any sense limiting.

What is claimed is:

1. A computer-implemented system for structuring a financial term definition, instantiating said financial term definition as a financial term, and evaluating said financial term, said system comprising:

a non-empty library (504) of term definitions (506) stored in the memory of said computer which includes compiled term definitions and zero or more previously structured financial term definitions;

means for choosing a financial term definition (56) from said library;

means for selecting a term definition from said library and instantiating, as a term, said selected term definition within said chosen financial term definition (57);

means for establishing communication between a plurality of such said terms thereby structuring said chosen financial term definition (60);

means for storing said chosen financial term definition in said library (66);

means for instantiating said chosen financial term definition as a financial term (57); and

means for evaluating said financial term to yield at least one selected value related to the financial term (712).

2. The system of claim 1, further comprising means for representing the plurality of such said terms with communications established therebetween as a new financial term definition (54), and means for instantiating said new financial term definition in a new network of such said terms with communications established therebetween (57).

3. The system of claim 1, wherein each said term has a type (710), and further comprising means for allowing establishment of communications between terms of compatible type and disallowing establishment of communications between terms of incompatible type (708).

4. The system of claim 1, wherein each said term has a type (710), and further comprising at least one said term to which first and second of other said terms are capable of having communication established thereto (706), further comprising means for allowing establishment of communication between said first and second other said terms

that are compatible and disallowing establishment of communication between said first and second other said terms to said at least one of said terms when the types of the first and second other said terms are incompatible (708).

5. The system for structuring a financial term definition of claim 1, further comprising a graphical user interface portion for selecting term definitions (50), instantiating said term definitions as terms (57), and for establishing a graphical connection between said terms representing the establishment of communication therebetween (60).

6. The system of claim 5, further comprising a display apparatus (50) having at least one window region in which said term definitions are instantiated (54).

7. The system of claim 1, wherein said financial term represents a financial instrument, and furthermore said means for evaluating said financial term is capable of yielding at least the net present value of said financial instrument (106).

8. The system of claim 1, wherein at least one of said terms includes private data capable of being passed for processing to other of said terms by way of said established communication, further comprising means for allowing modification of said private data by a user with access authority and for preventing modification of said private data by a user without access authority (224).

9. The system of claim 1, further including means for modifying said chosen term definition (60).

10. The system of claim 1, wherein said means for establishing communication between a plurality of such said terms represents means for said financial term to operate asynchronously as a plurality of intercommunicating computer processes.

11. In a computer-implemented system for structuring a financial term definition, instantiating said financial term definition as a financial term, and evaluating said financial term, wherein said system comprises a non-empty library of term definitions

stored in the memory of said computer and contains compiled term definitions and zero or more previously structured financial term definitions, a method comprising the steps of:

choosing a financial term definition from said library (56);

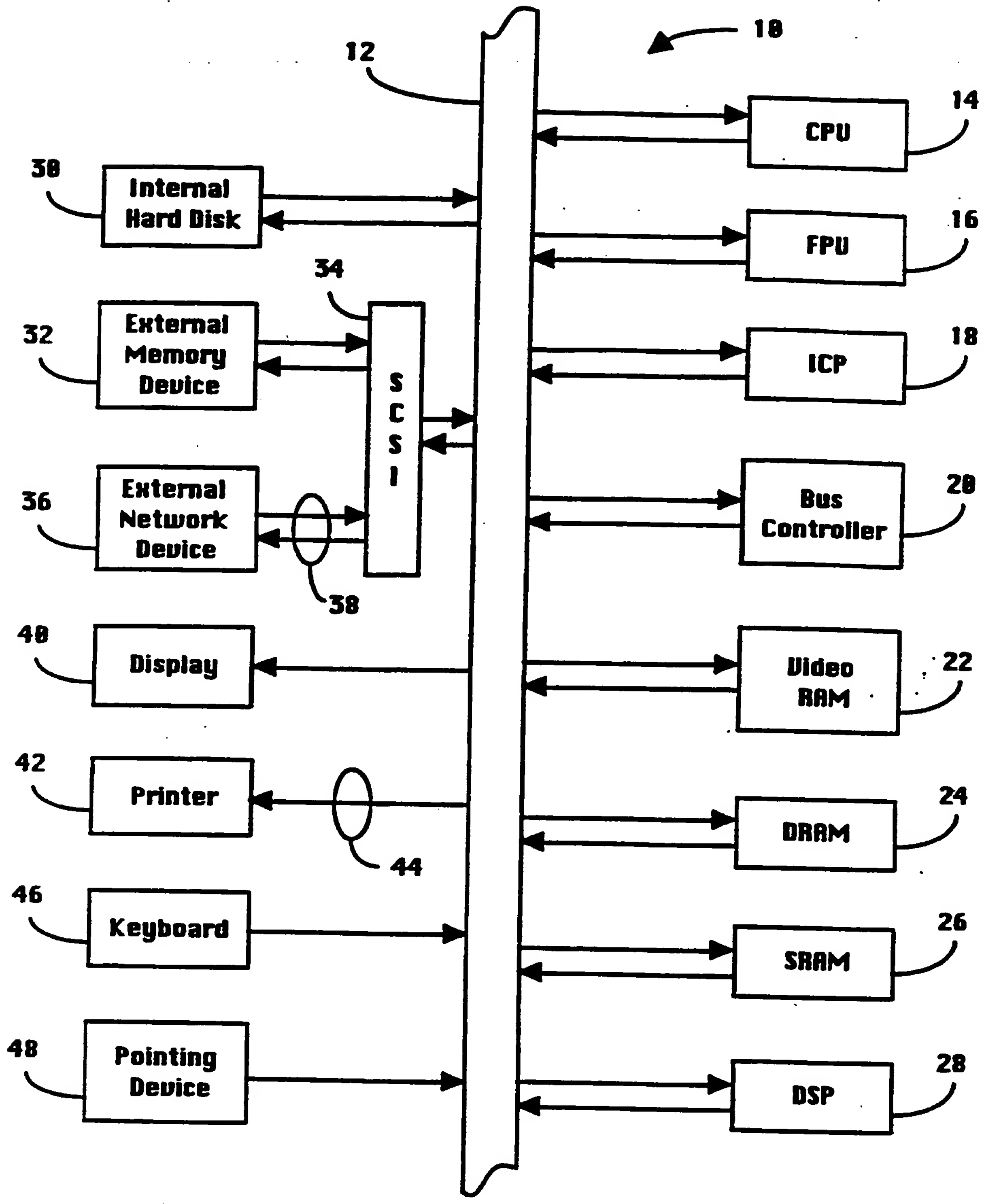
selecting a term definition from said library and instantiating, as a term, said selected term definition within said chosen financial term definition (57);

establishing communication between a plurality of such said terms thereby structuring said chosen financial term definition (60);

storing said chosen financial term definition in said library (66);

instantiating said chosen term definition as a financial term (57); and

evaluating said financial term to yield at least one selected value related to the financial term (712).



10
12
14
CPU
16
FPU
18
ICP
20
Bus Controller
22
Video RAM
24
DRAM
26
SRAM
28
DSP
30
Internal Hard Disk
32
External Memory Device
34
SCSI
36
External Network Device
38
40
Display
42
Printer
44
46
Keyboard
48
Pointing Device

Fig. 1

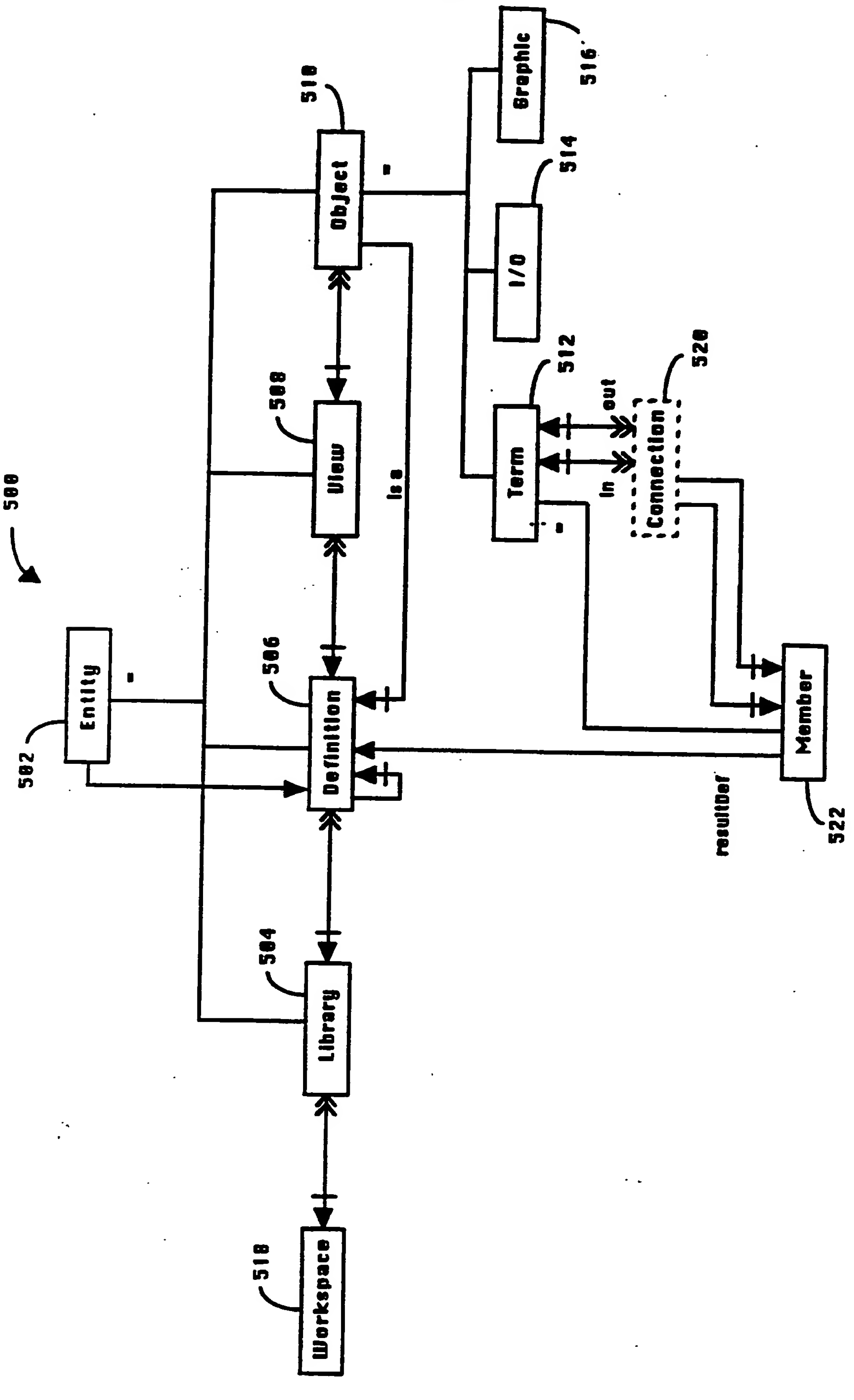

500
Entity
502
Library
504
Definition
506
View
508
Object
510
Workspace
518
Term
512
I/O
514
Graphic
516
Connection
520
Member
522
in
out
is a
resultDef

Fig. 2

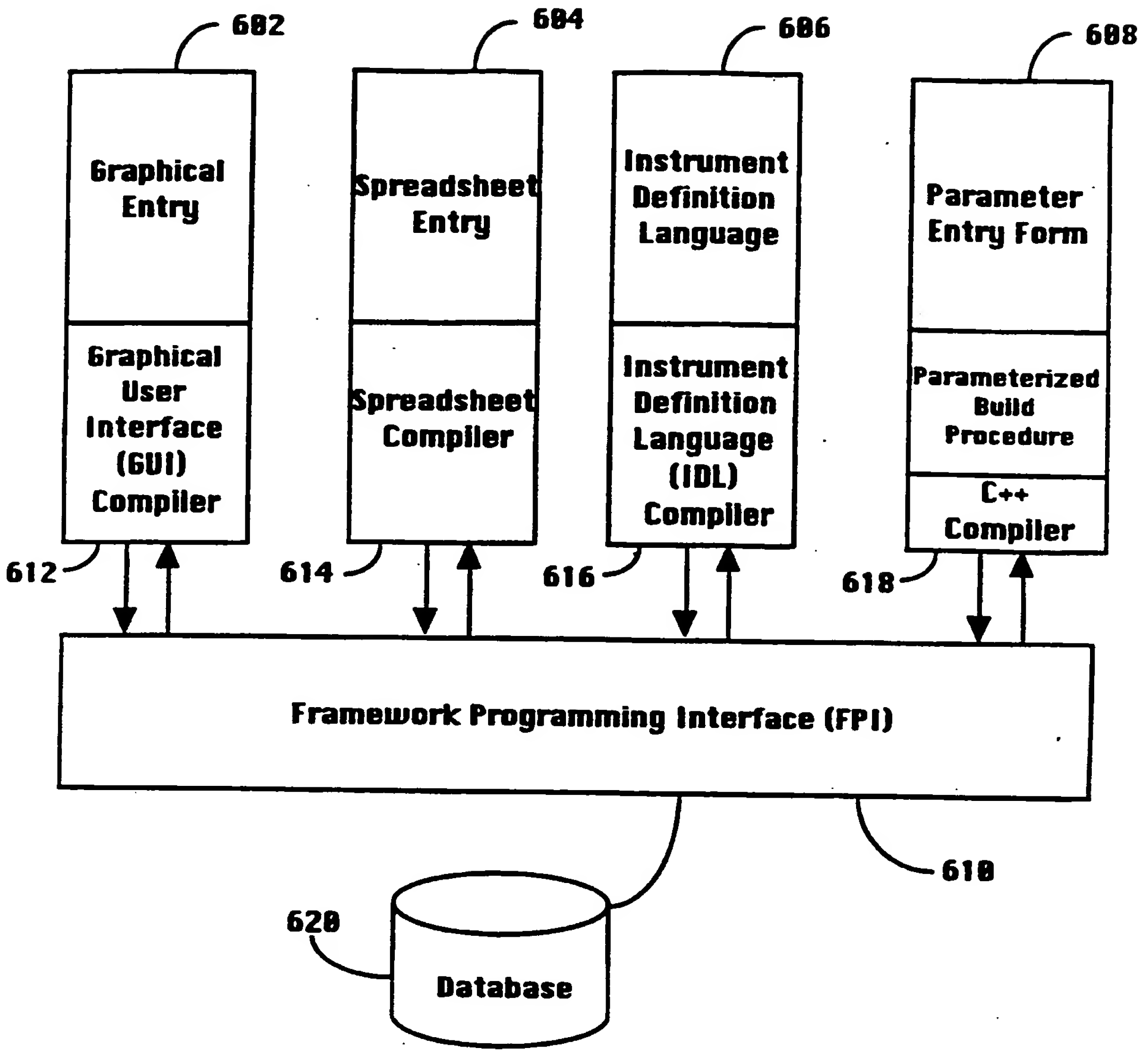

602
604
606
608
Graphical Entry
Spreadsheet Entry
Instrument Definition Language
Parameter Entry Form
Graphical User Interface (GUI) Compiler
Spreadsheet Compiler
Instrument Definition Language (IDL) Compiler
Parameterized Build Procedure
C++ Compiler
612
614
616
618
Framework Programming Interface (FPI)
610
620
Database

Fig. 3

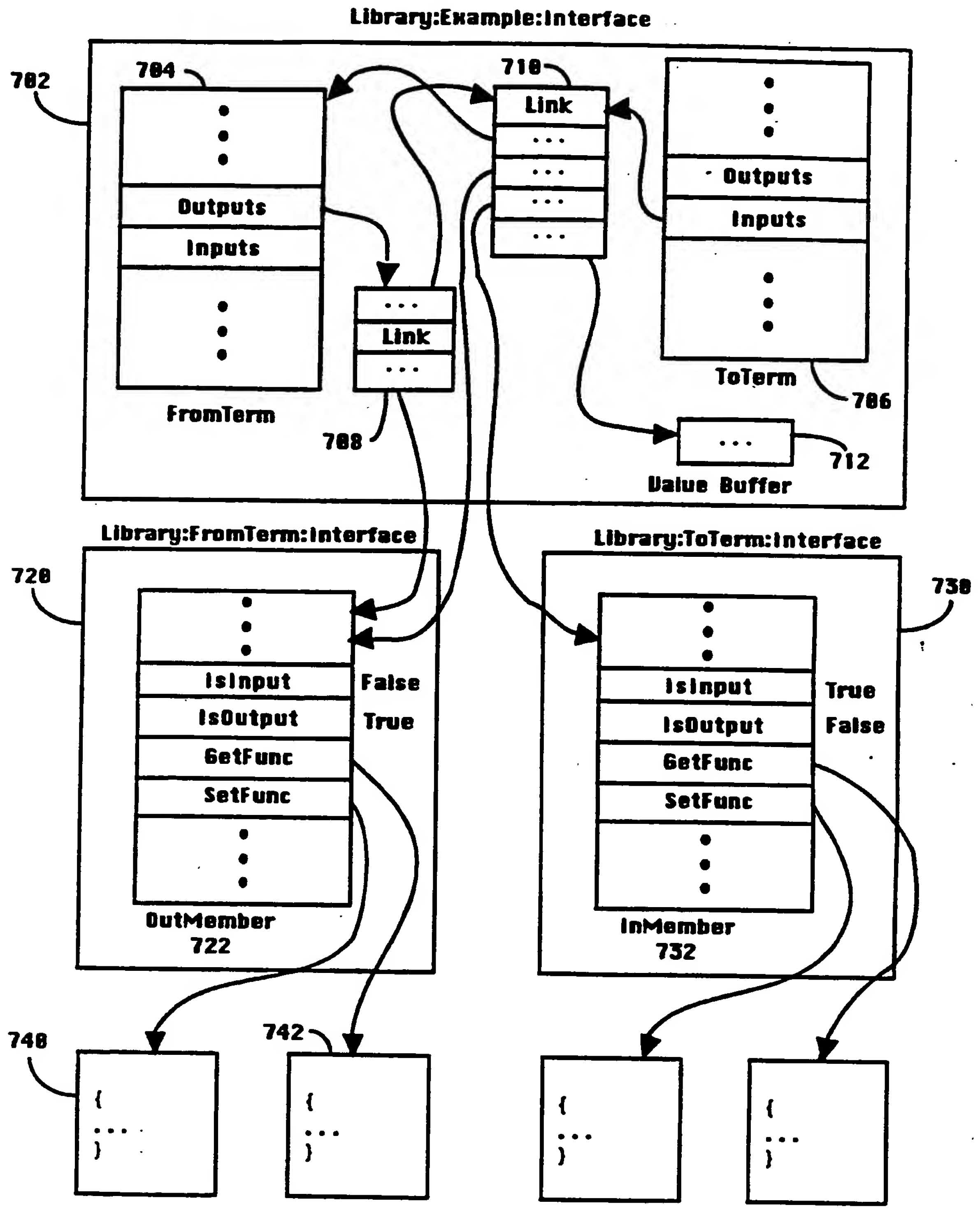

Library:Example:Interface
702
704
710
Link
...
...
...
...
Outputs
Inputs
Outputs
Inputs
...
Link
...
FromTerm
708
ToTerm
706
...
712
Value Buffer
Library:FromTerm:Interface
Library:ToTerm:Interface
720
730
IsInput
False
IsOutput
True
GetFunc
SetFunc
OutMember
722
IsInput
True
IsOutput
False
GetFunc
SetFunc
InMember
732
740
742
{ ... }
{ ... }
{ ... }
{ ... }

Fig. 4

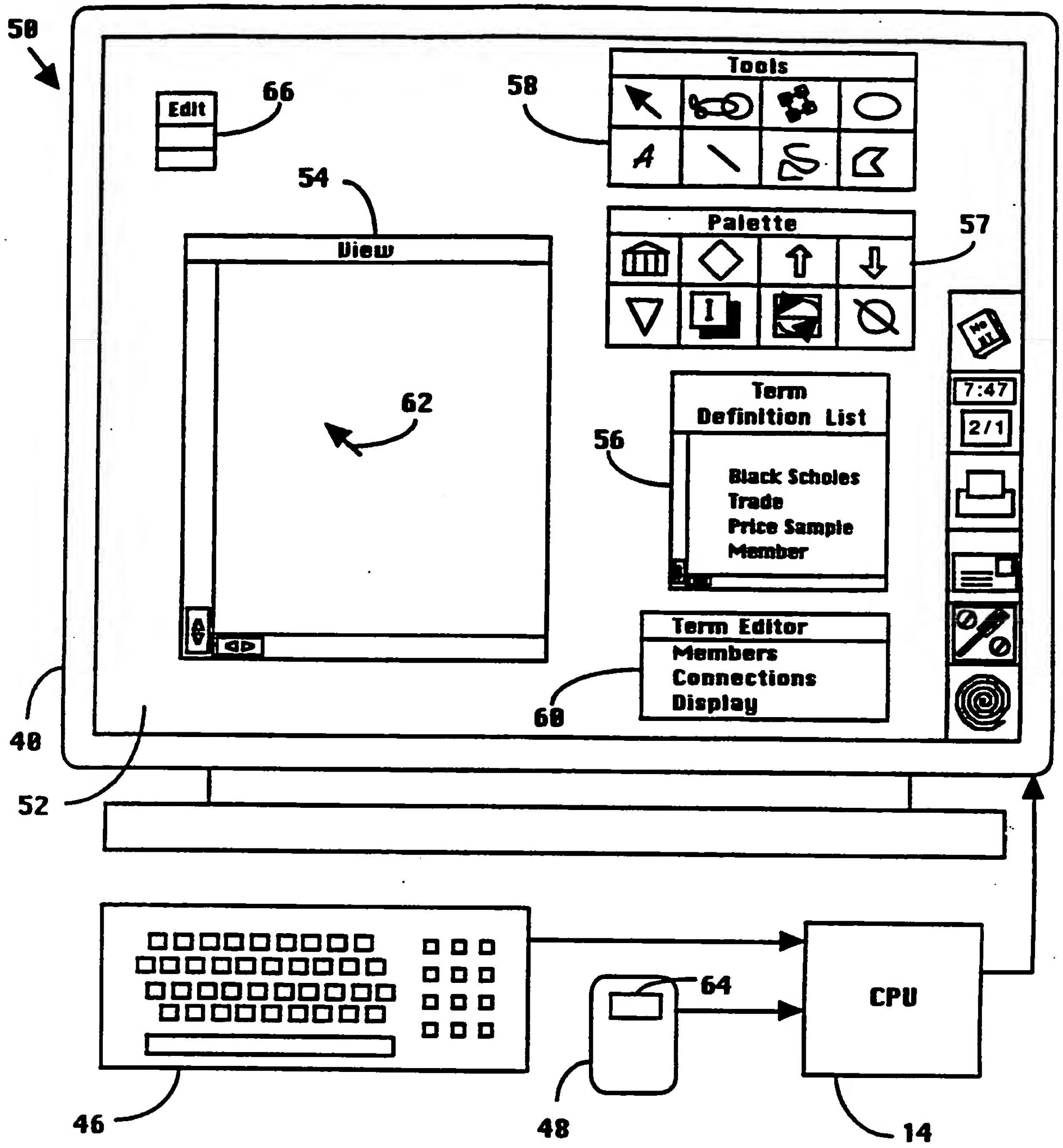

50
Edit
66
58
Tools
54
View
Palette
57
62
Term Definition List
56
Black Scholes
Trade
Price Sample
Member
7:47
2/1
Term Editor
Members
Connections
Display
60
40
52
64
CPU
46
48
14

Fig. 5

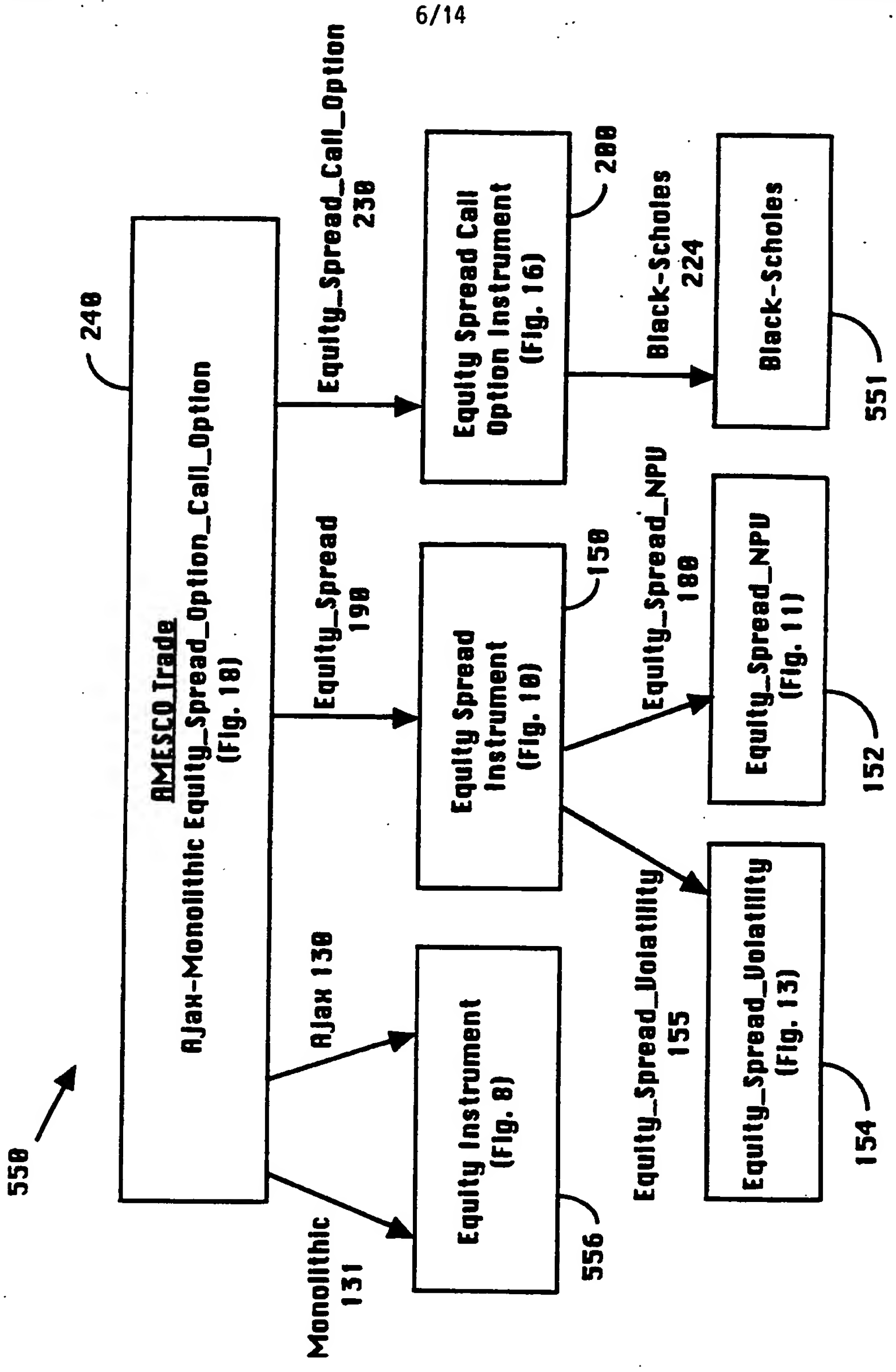
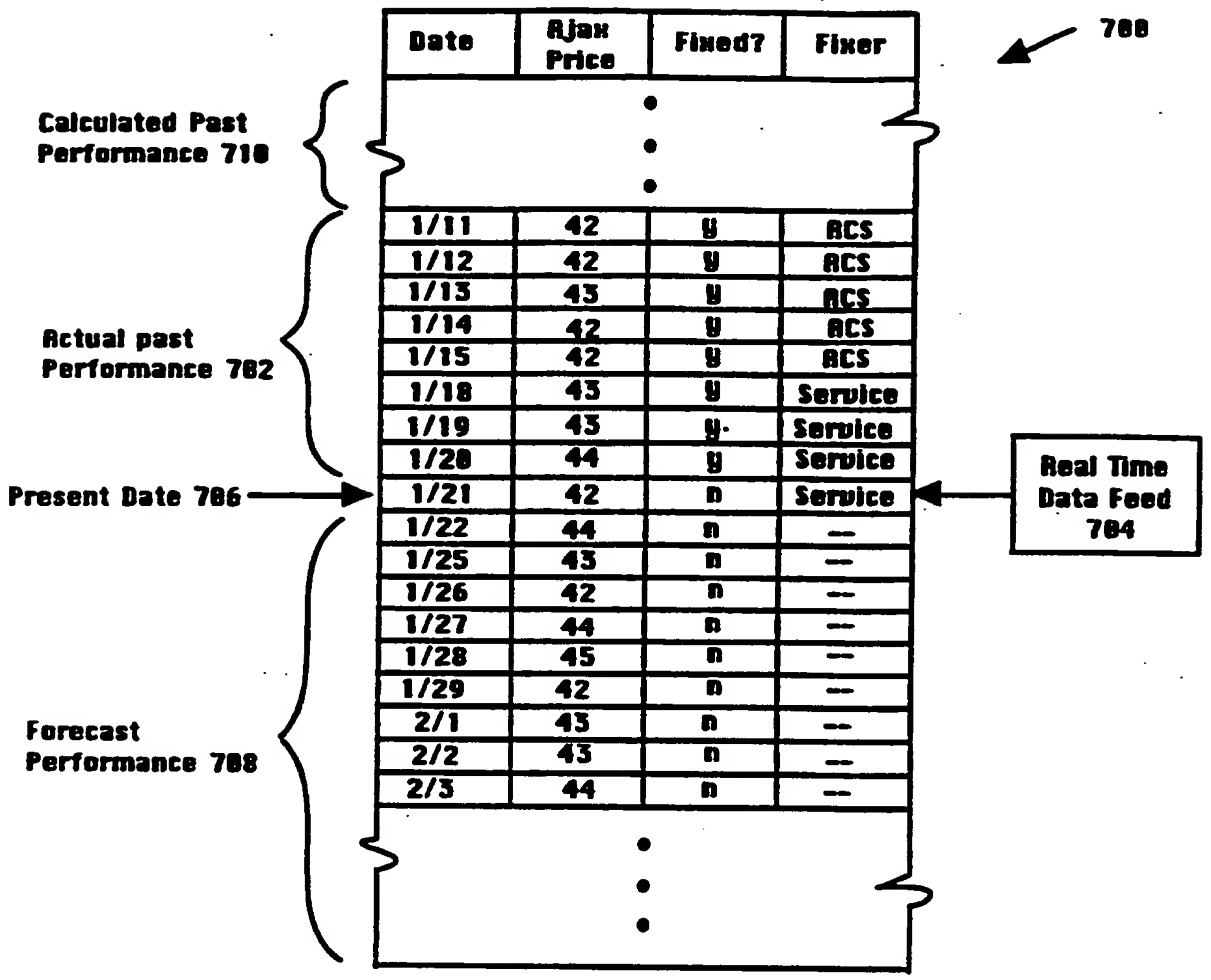

550
240
AMESCO Trade
AJax-Monolithic Equity_Spread_Option_Call_Option
(Fig. 18)
Monolithic 131
AJax 130
Equity_Spread 190
Equity_Spread_Call_Option 230
Equity Instrument (Fig. 8)
556
Equity Spread Instrument (Fig. 10)
150
Equity Spread Call Option Instrument (Fig. 16)
200
Equity_Spread_Volatility 155
Equity_Spread_NPV 180
Black-Scholes 224
Equity_Spread_Volatility (Fig. 13)
154
Equity_Spread_NPV (Fig. 11)
152
Black-Scholes
551

Fig. 6



Date	Ajax Price	Fixed?	Fixer
•			
•			
•			
1/11	42	y	ACS
1/12	42	y	ACS
1/13	43	y	ACS
1/14	42	y	ACS
1/15	42	y	ACS
1/18	43	y	Service
1/19	43	y	Service
1/20	44	y	Service
1/21	42	n	Service
1/22	44	n	—
1/25	43	n	—
1/26	42	n	—
1/27	44	n	—
1/28	45	n	—
1/29	42	n	—
2/1	43	n	—
2/2	43	n	—
2/3	44	n	—
•			
•			
•			

Fig. 7

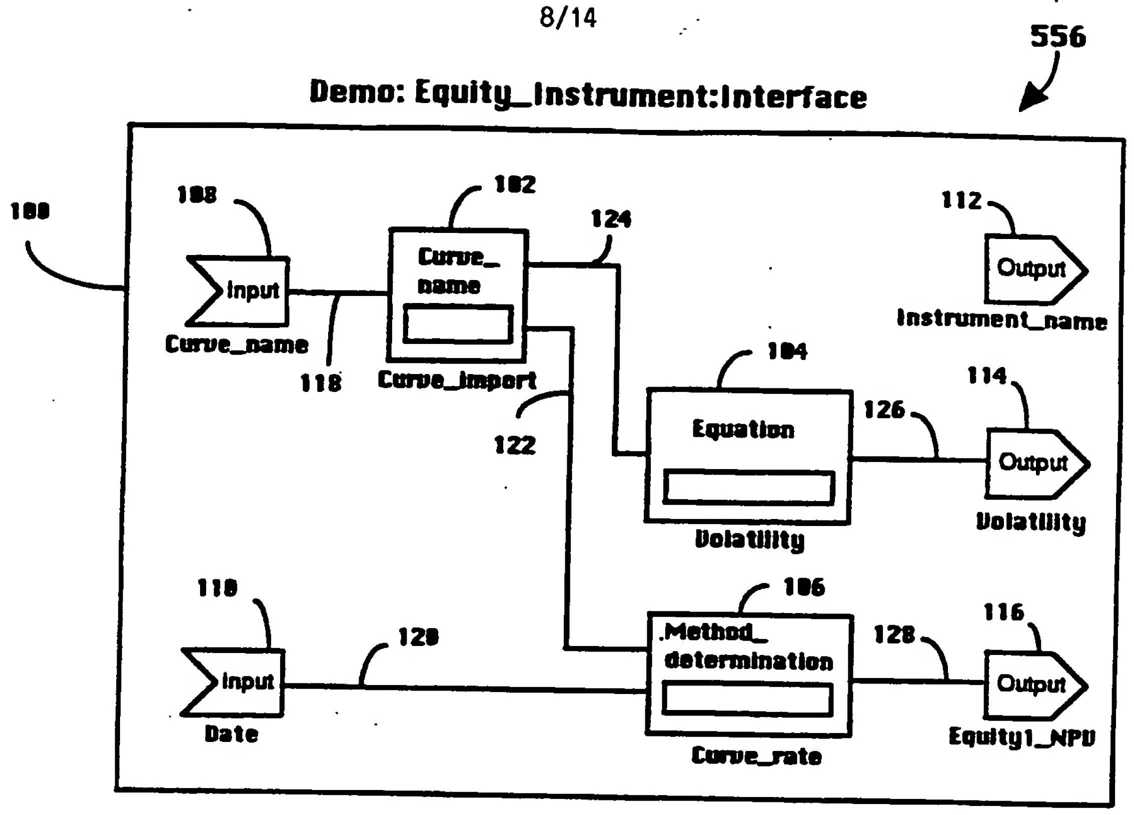

556
Demo: Equity_Instrument:Interface
100
108
Input
Curve_name
118
102
Curve_name
Curve_import
124
112
Output
Instrument_name
104
Equation
Volatility
122
126
114
Output
Volatility
110
Input
Date
120
106
Method_determination
Curve_rate
128
116
Output
Equity1_NPV

Fig. 8

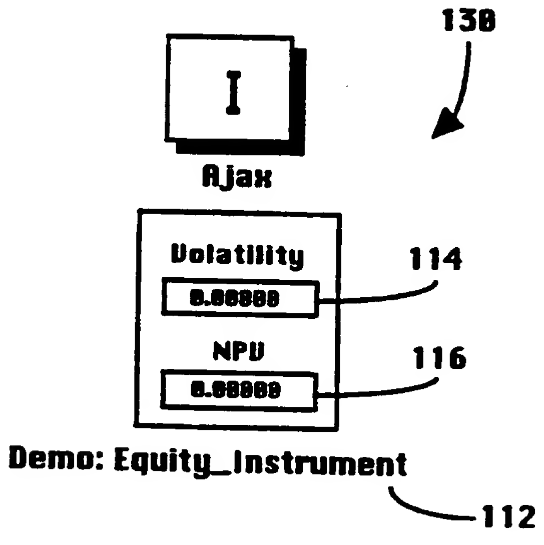

130
I
Ajax
Volatility
0.00000
114
NPV
0.00000
116
Demo: Equity_Instrument
112

Fig. 9

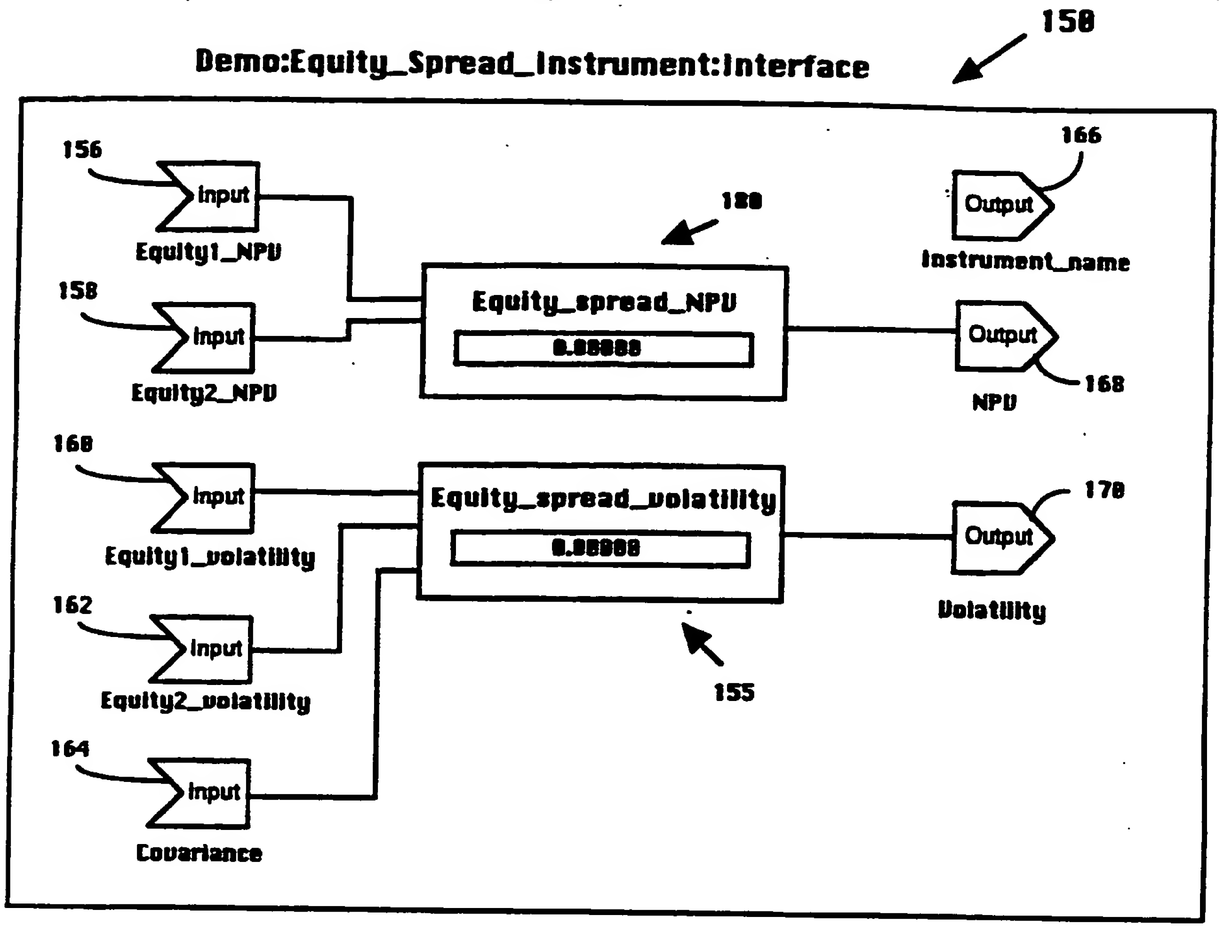

150
Demo:Equity_Spread_Instrument:Interface
156
Input
Equity1_NPV
158
Input
Equity2_NPV
180
Equity_spread_NPV
0.00000
166
Output
Instrument_name
Output
NPV
168
160
Input
Equity1_volatility
162
Input
Equity2_volatility
164
Input
Covariance
Equity_spread_volatility
0.00000
155
170
Output
Volatility

Fig. 10

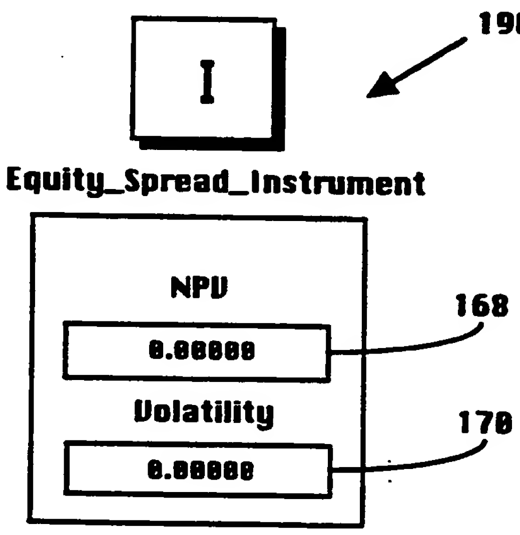

190
I
Equity_Spread_Instrument
NPV
0.00000
168
Volatility
0.00000
170

Fig. 15

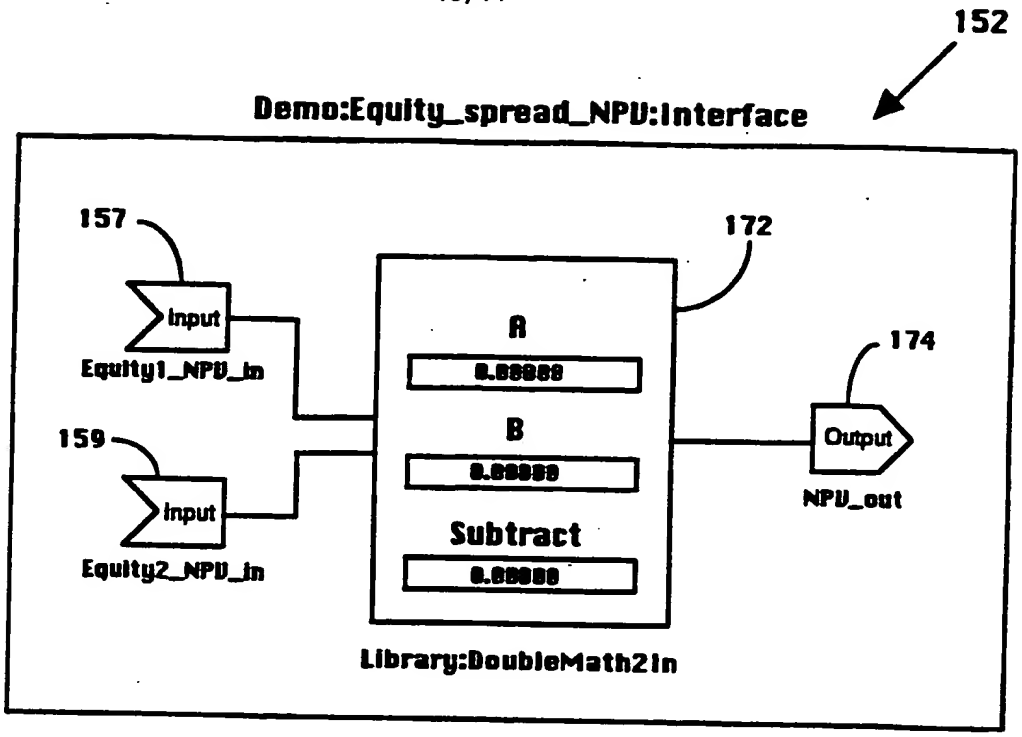

152
Demo:Equity_spread_NPV:Interface
157
Input
Equity1_NPV_in
159
Input
Equity2_NPV_in
172
A
0.00000
B
0.00000
Subtract
0.00000
Library:DoubleMath2In
174
Output
NPV_out

Fig. 11

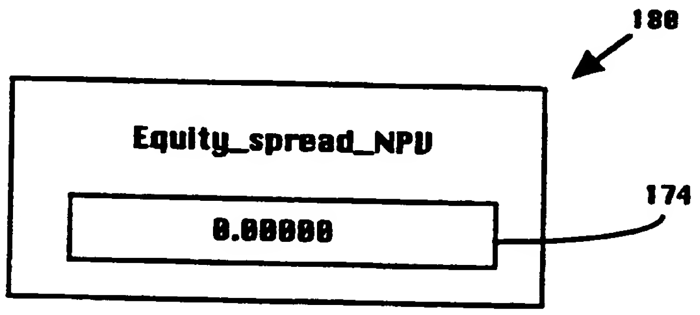

188
Equity_spread_NPV
0.00000
174

Fig. 12

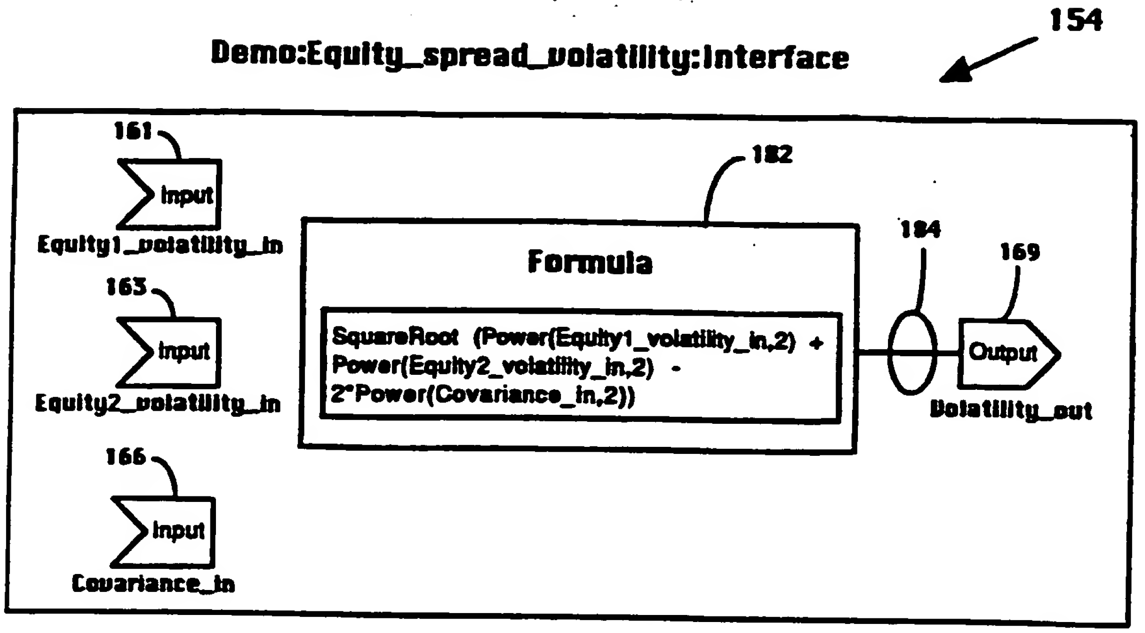

154
Demo:Equity_spread_volatility:Interface
161
Input
Equity1_volatility_in
163
Input
Equity2_volatility_in
166
Input
Covariance_in
182
Formula
SquareRoot (Power(Equity1_volatility_in,2) + Power(Equity2_volatility_in,2) - 2*Power(Covariance_in,2))
184
169
Output
Volatility_out

Fig. 13

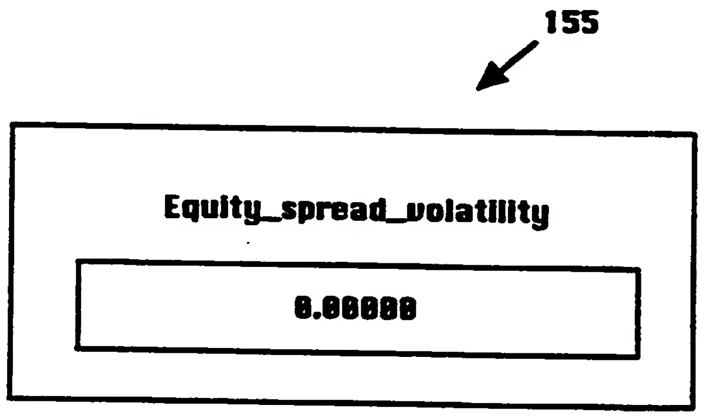

155
Equity_spread_volatility
0.00000

Fig. 14

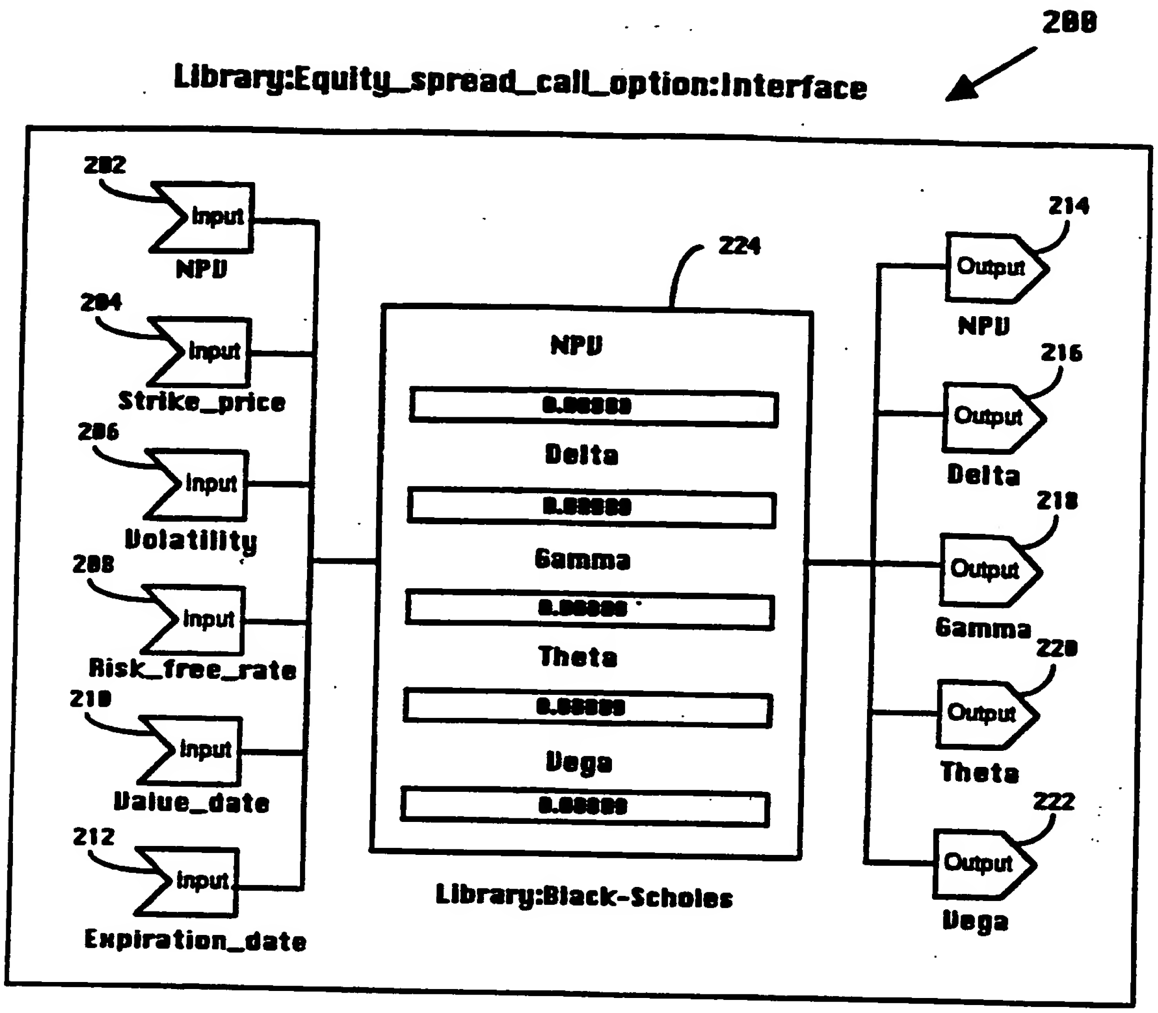

200
Library:Equity_spread_call_option:Interface
202
Input
NPV
204
Input
Strike_price
206
Input
Volatility
208
Input
Risk_free_rate
210
Input
Value_date
212
Input
Expiration_date
224
NPV
0.00000
Delta
0.00000
Gamma
0.00000
Theta
0.00000
Vega
0.00000
Library:Black-Scholes
214
Output
NPV
216
Output
Delta
218
Output
Gamma
220
Output
Theta
222
Output
Vega

Fig. 16

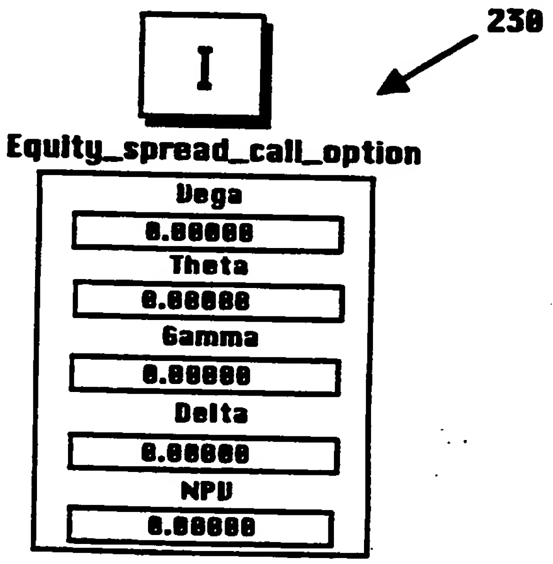

230
I
Equity_spread_call_option
Vega
0.00000
Theta
0.00000
Gamma
0.00000
Delta
0.00000
NPV
0.00000

Fig. 17

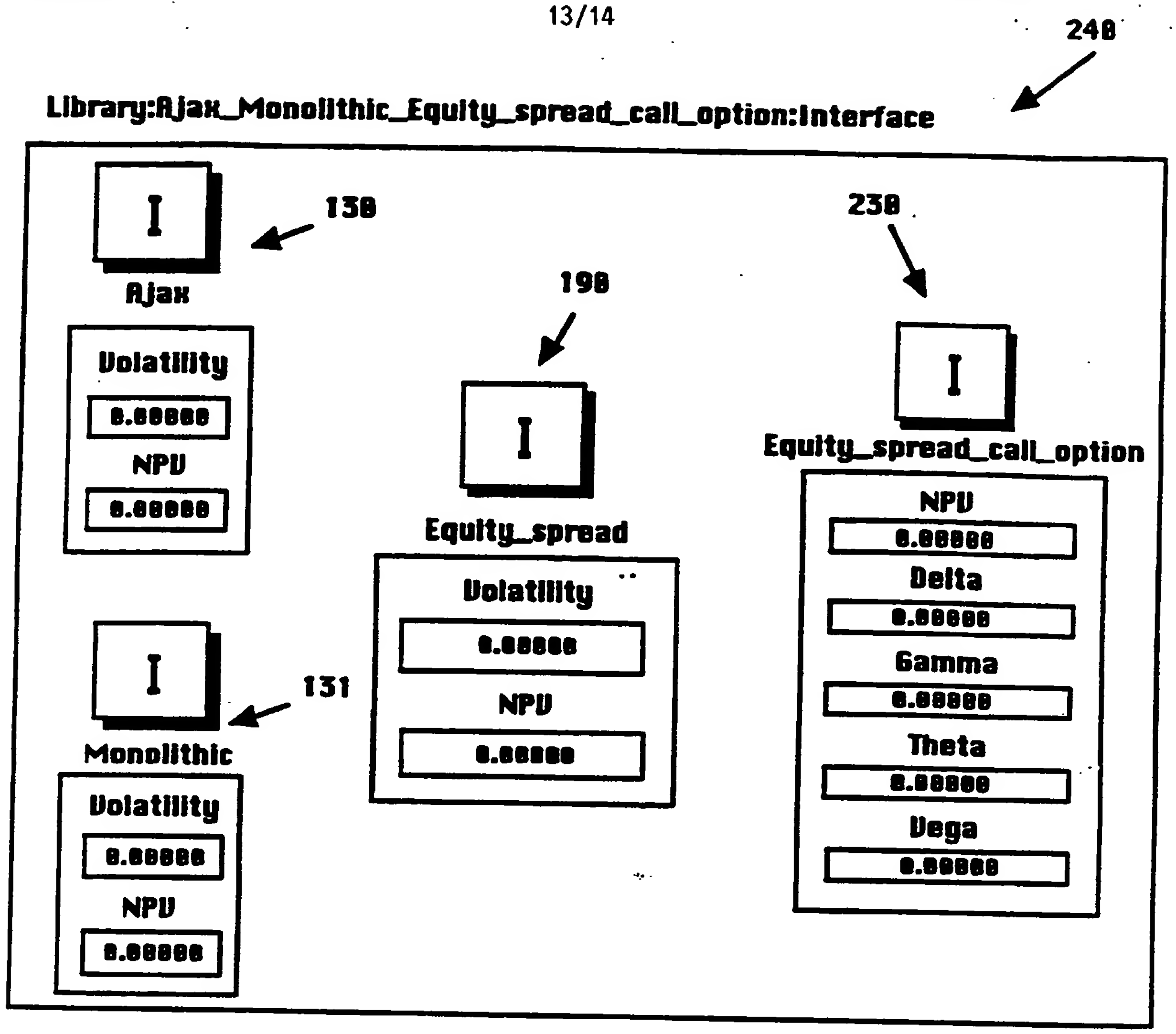

240
Library:Ajax_Monolithic_Equity_spread_call_option:Interface
130
Ajax
Volatility
0.00000
NPV
0.00000
190
Equity_spread
Volatility
0.00000
NPV
0.00000
230
Equity_spread_call_option
NPV
0.00000
Delta
0.00000
Gamma
0.00000
Theta
0.00000
Vega
0.00000
131
Monolithic
Volatility
0.00000
NPV
0.00000

Fig. 18

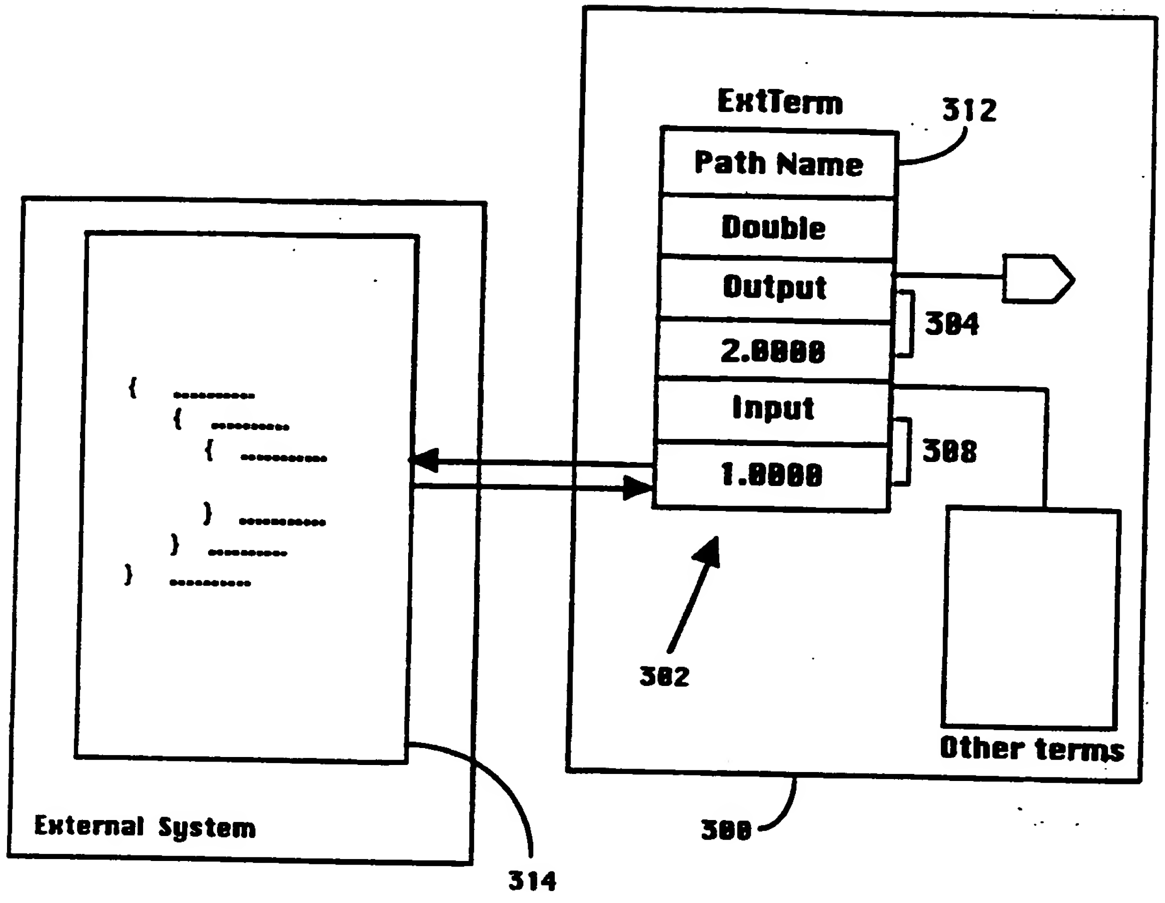

ExtTerm
312
Path Name
Double
Output
304
2.0000
Input
308
1.0000
302
Other terms
300
External System
314

Fig. 19

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 94/02468

A. CLASSIFICATION OF SUBJECT MATTER
IPC 5 G06F15/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ASSOCIATION FOR COMPUTING MACHINERY - SIGPLAN NOTICES vol. 27, no. 10 , October 1992 , NEW YORK, US pages 166 - 177 XP000327296 T. EGGENSCHWILER ET AL 'ET++SwapsManager: Using Object Technology in the Financial Engineering Domain' see page 170, right column, paragraph 1 - page 175, left column, paragraph 3 --- -/--	1,2,5-9, 11

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

2

Date of the actual completion of the international search 29 June 1994	Date of mailing of the international search report 05.07.94
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Pottiez, M

INTERNATIONAL SEARCH REPORT

Inter. nal Application No
PCT/US 94/02468

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	PROCEEDINGS OF THE 7TH CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS February 1991 , MIAMI, US pages 168 - 174 XP000298920 M. BENAROCH ET AL 'An Intelligent Assistant for Financial Hedging' see page 170, left column, paragraph 2 - right column, paragraph 3 see page 172, left column, paragraph 6 - right column, paragraph 2 ---	1
P,X	INFORMATION AND MANAGEMENT vol. 24, no. 5 , May 1993 , NETHERLANDS pages 267 - 281 A. BANSAL ET AL 'Financial Risk and Financial Risk Management Technology (RMT)' see page 272, last paragraph - page 274, paragraph 1 ---	1-11
A	PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE ON WALL STREET - NEW YORK 9 October 1991 , LOS ALAMITOS, US pages 73 - 78 XP000400726 T.M. PATEL ET AL 'The SPLENDORS Real Time Portfolio Management System' see page 73, right column see page 74, right column, paragraph 3 - page 76, left column, paragraph 5 ---	1-11
A	WALL STREET COMPUTER REVIEW vol. 6, no. 6 , March 1989 , US pages 42, 43, 45, 46, 48, 50, 80, 81 L. KOFLOWITZ 'Hedging Tools provide Portfolio Security Blanket' see the whole document ---	1,7,10, 11
A	EP,A,0 294 187 (CORPORATE CLASS SOFTWARE) 7 December 1988 see abstract -----	

2

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter   nal Application No
PCT/US 94/02468

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0294187	07-12-88	US-A-	4989141	29-01-91
		AU-A-	1695288	01-12-88
		US-A-	5189608	23-02-93